Project4-demo

Yuhan Sun

April 4, 2016

Prepare data

review/helpfulness: fraction of users who found the review helpful review/score: rating of the product review/time: time of the review (unix time)

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
load('/Users/sunxiaohan/Desktop/project4/data_new.Rdata')
load('/Users/sunxiaohan/Desktop/project4/user.table.Rdata')
load('/Users/sunxiaohan/Desktop/project4/product.table.Rdata')
dim(data_new)
```

```
## [1] 505190     20
```

```
dim(user.table)
```

```
## [1] 3490    5
```

```
dim(product.table)
```

```
## [1] 4250    4
```

```
#calculate the sd
data_new[,'dif']=(data_new$review_score-data_new$PReview_ave)^2

# summarise sd for each individual user
user.sd=data_new%>%
                group_by(review_userid)%>%
                summarize(
                  sd_total=mean(dif,na.rm=T)
                )

user.table=left_join(user.table,user.sd,by='review_userid')
user.table=na.omit(user.table)
user.table=user.table[order(user.table$sd_total),]


word_for_user=data_new%>%
  group_by(review_userid)%>%
  summarize(
    word_ave=mean(word_count)
  )
```

Find Connoisseurs

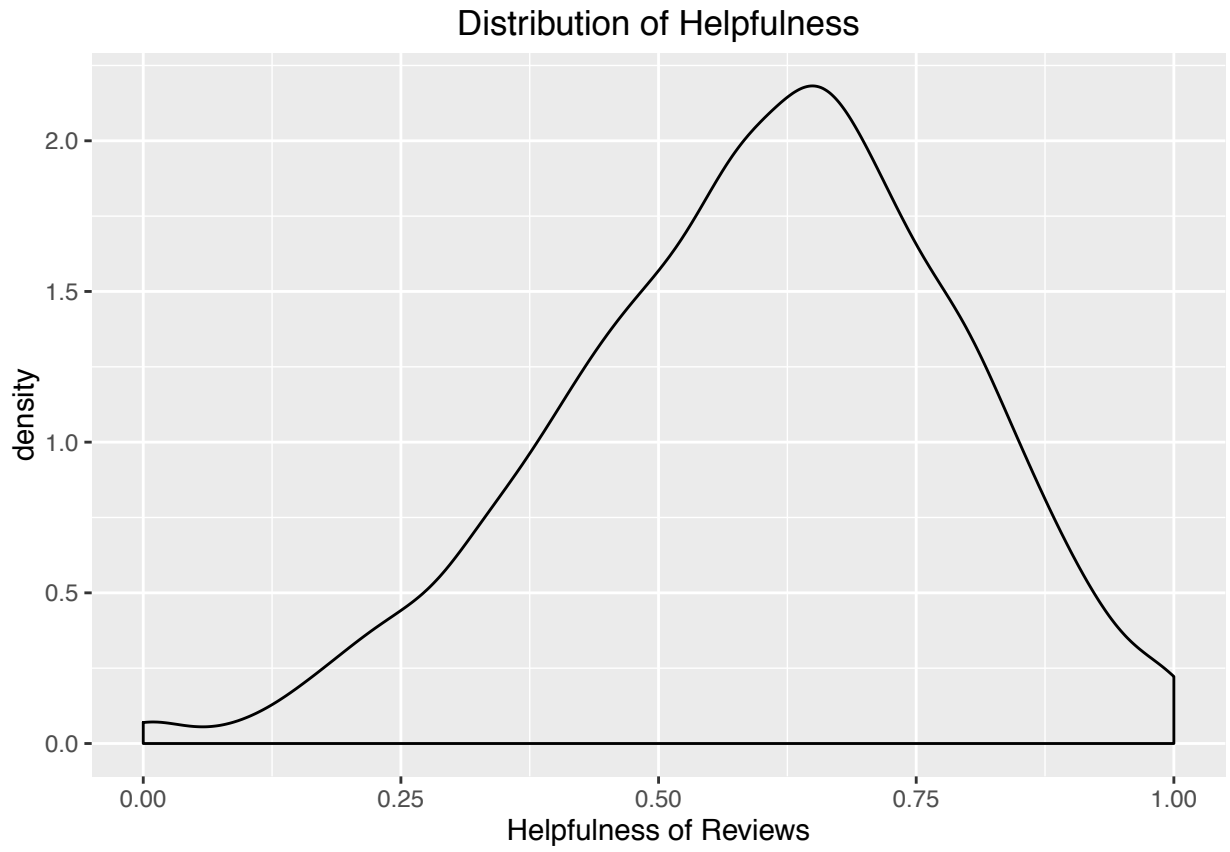
Critiria

```
* reviews have over 5 votes
* reviews have over 0.6 helpfulness ratio
* user need to write at least 5 reviews
* product should have at least 100 reviews
* calculate the variance of it to the overall variance and choose the top 500
```

```
# user should write at least 5 reviews
user.filter=filter(user.table,user.count>5)
dim(user.filter)
```

```
## [1] 3484    6
```

```
#distribution of helpfulness
ggplot()+geom_density(aes(user.table$UReview_help))+xlab('Helpfulness of Reviews')+ggtitle('Distributio
```




```
quantile(user.table$UReview_help,0.60)
```

```
##       60%
## 0.6595542
```

```
# Select 60% quantile as the cutting point
user.filter=filter(user.filter,UReview_help>0.6)
dim(user.filter)
```

```
## [1] 1834    6
```

```
# At least 5 user read this review
user.filter=filter(user.filter,UReview_read>5)
dim(user.filter)
```

```
## [1] 757   6
```

```
# Choose the top 500 as Connoisseurs
connoi=user.filter[1:500,]
#connoi=left_join(connoi,user.table,by='review_userid')
#save(connoi,file='connoi.Rdata')
```

Find Extreme Case

Criteria

* User should write at least 5 reviews
* The helpfulness of reviews should be smaller than 0.4

```
extreme.raw=user.table[order(user.table$sd_total,decreasing = T),]
dim(extreme.raw)
```

```
## [1] 3484    6
```

```
# user should write at least 10 reviews
extreme=filter(extreme.raw,user.count>5)
dim(extreme)
```

```
## [1] 3484    6
```

```
# the helpfulness of reviews should be at most 0.4
#quantile(user.table$UReview_help,0.40)
extreme=filter(extreme,UReview_help<0.4)
dim(extreme)
```

```
## [1] 538   6
```

```
# choose the top 500 as extreme case
extreme=extreme[1:500,]
head(extreme)
```

```
## Source: local data frame [6 x 6]
##
##    review_userid user.count UReview_ave UReview_read UReview_help
##            (chr)      (int)       (dbl)        (dbl)        (dbl)
## 1 A2YM6JTQIBZ8YC         54    1.222222     42.09259    0.1971570
## 2  ASU5IH3CM6XXE        115    1.486957     39.83478    0.2308785
## 3 A171WA0E3DIXXM         61    1.000000     29.59016    0.3539053
## 4 A3R32VYVC8IJB9         60    1.600000     22.13333    0.2579220
## 5 A24PA468O7ED7J         55    1.145455     16.36364    0.2192424
## 6 A1ECVYFEXREVC7         59    1.254237     14.94915    0.1923540
## Variables not shown: sd_total (dbl)
```

```
#save(extreme,file='extreme.Rdata')
```

Find the Amateurs

Critetia

Amateurs=General - Connoisseurs

```
find1=left_join(user.table,connoi,by='review_userid')
find1[,'index']=seq(1:nrow(find1))
find2=filter(find1,user.count.y!='NA')
ame=user.table[-find2$index,]
head(ame)
```

```
## Source: local data frame [6 x 6]
##
##    review_userid user.count UReview_ave UReview_read UReview_help
##            (chr)      (int)       (dbl)        (dbl)        (dbl)
## 1 A204D78BPJEF2W         64    5.000000   0.78125000    0.9166667
## 2 A2SRPX9AZVAMWT         70    5.000000   0.05714286    0.5000000
## 3 A2YIHBD7MPNTC9        112    5.000000   0.16071429    0.4166667
## 4 A3FVISMTESSRUL        234    5.000000   0.11111111    1.0000000
## 5  AX52ULYSK82AF         57    4.192982   1.08771930    0.7187500
## 6 A2IKWMHSHKRLGG         54    4.851852   1.25925926    0.6333333
## Variables not shown: sd_total (dbl)
```

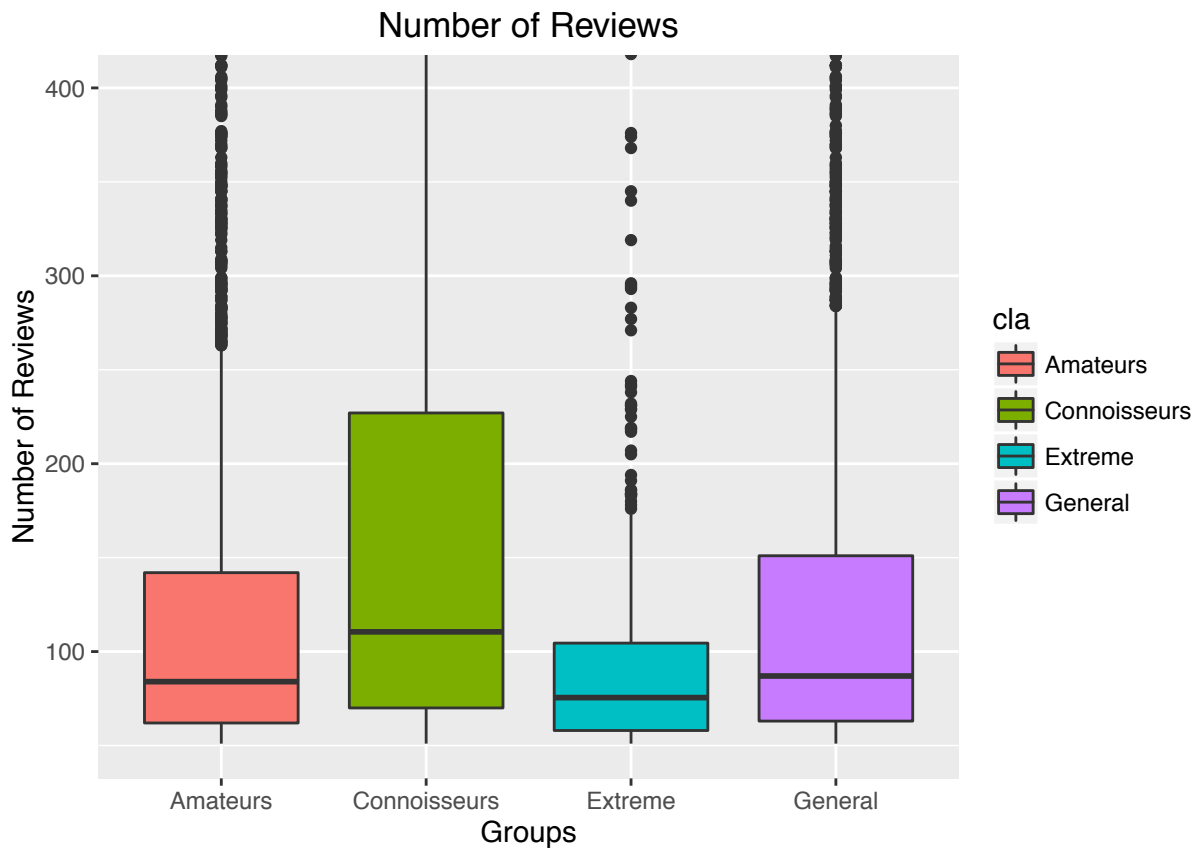
Add the number of words

```
connoi=left_join(connoi,word_for_user,by='review_userid')
ame=left_join(ame,word_for_user,by='review_userid')
extreme=left_join(extreme,word_for_user,by='review_userid')
user.table=left_join(user.table,word_for_user,by='review_userid')

con1=connoi
con1['cla']='Connoisseurs'
ame1=ame
ame1['cla']='Amateurs'
ext1=extreme
ext1['cla']='Extreme'
gen1=user.table
gen1['cla']='General'

whole=rbind(con1,ame1,ext1,gen1)
```

Comparasion between General, Connoisseurs, Amateurs and Extreme Case

Number of reviews

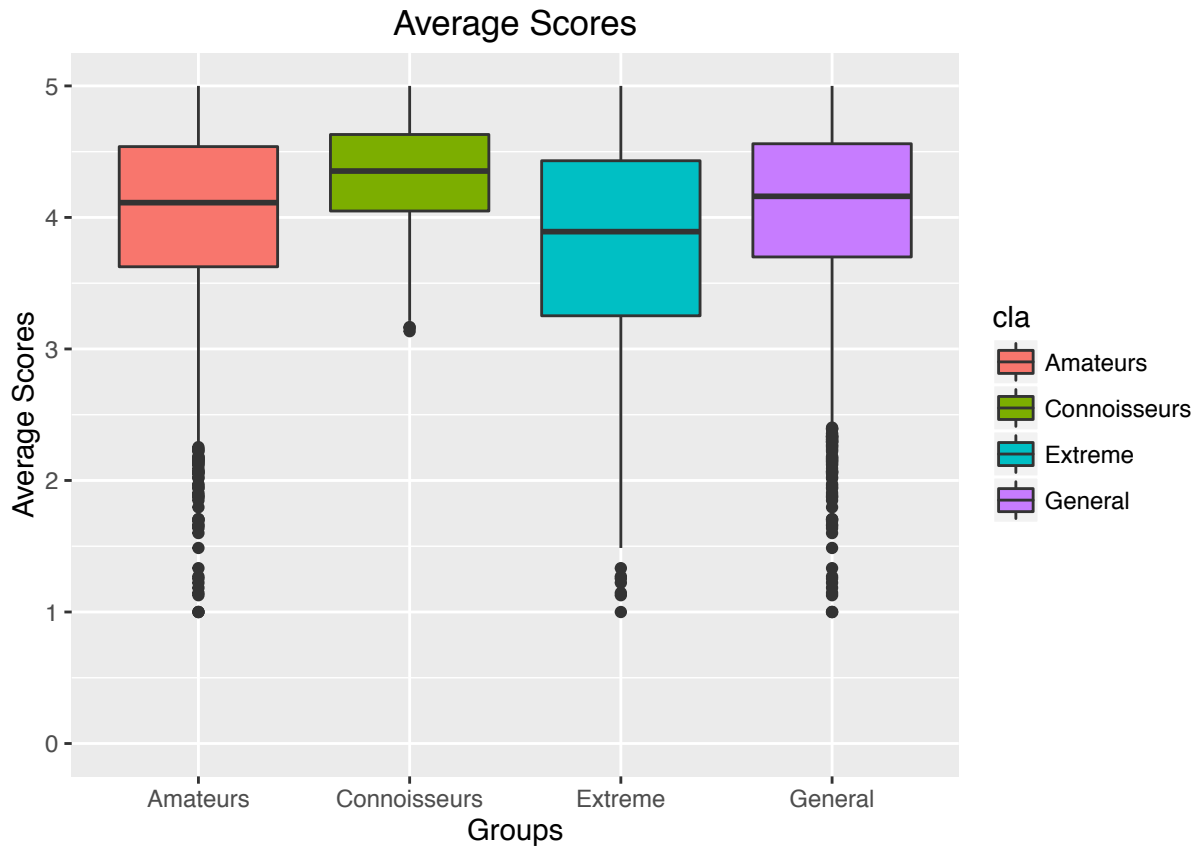
```
class=factor(whole$cla)
ggplot(whole,aes(cla,user.count))+geom_boxplot(aes(fill=cla))+coord_cartesian(ylim = c(50, 400))+
  xlab('Groups')+ylab('Number of Reviews')+ggtitle('Number of Reviews')
```




```
#par(mfrow=c(1,4))
#boxplot(connoi$user.count,ylim=c(0,200),main='Connoisseurs')
#boxplot(user.table$user.count,ylim=c(0,200),main='General')
#boxplot(ame$user.count,ylim=c(0,200),main='Amateurs')
```

Average score by individual

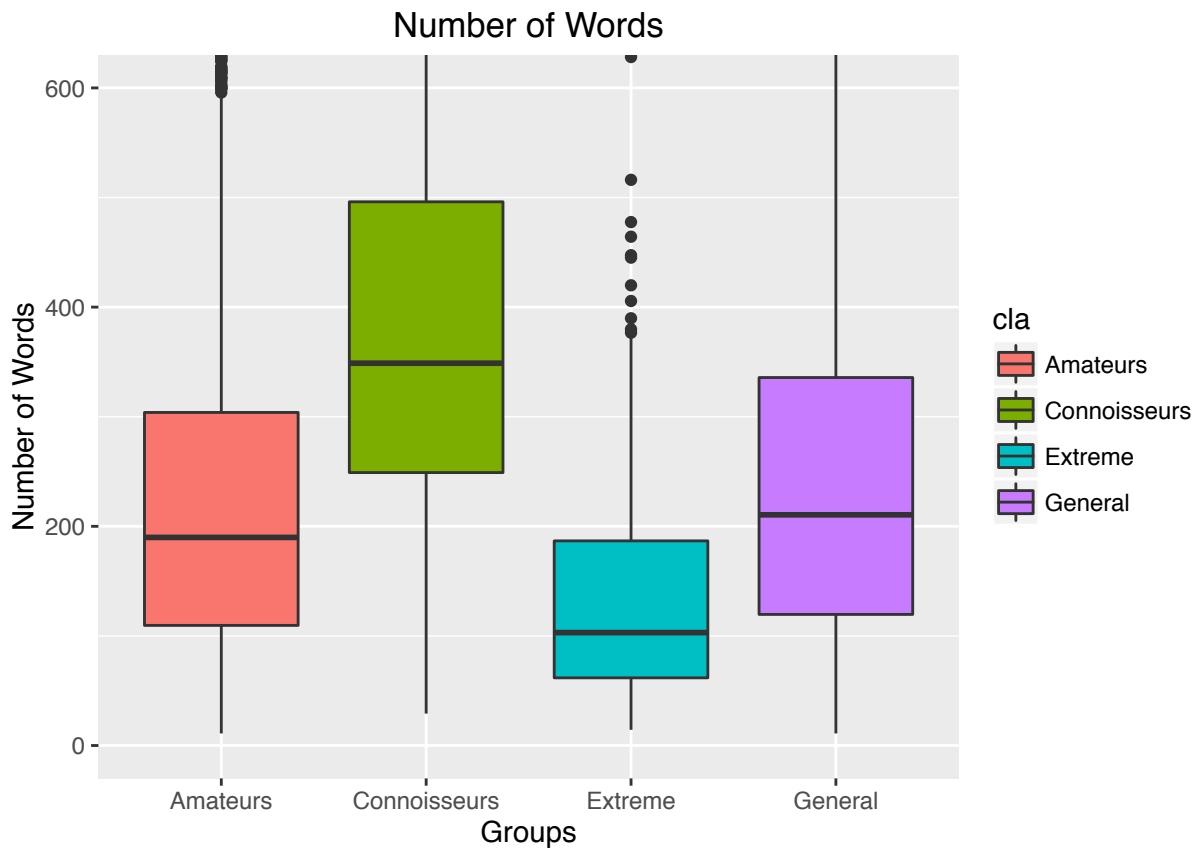
```
ggplot(whole,aes(cla,UReview_ave))+geom_boxplot(aes(fill=cla))+coord_cartesian(ylim = c(0, 5))+
  xlab('Groups')+ylab('Average Scores')+ggtitle('Average Scores')
```




```
#par(mfrow=c(1,3))
#boxplot(connoi$UReview_ave,main='Connoisseurs',ylim=c(0,5))
#boxplot(user.table$UReview_ave,main='General',ylim=c(0,5))
#boxplot(ame$UReview_ave,main='Amateurs',ylim=c(0,5))
#summary(connoi$UReview_ave)
#summary(user.table$UReview_ave)
```

Compare of the word_count

```
ggplot(whole,aes(cla,word_ave))+geom_boxplot(aes(fill=cla))+coord_cartesian(ylim = c(0, 600))+
  xlab('Groups')+ylab('Number of Words')+ggtitle('Number of Words')
```

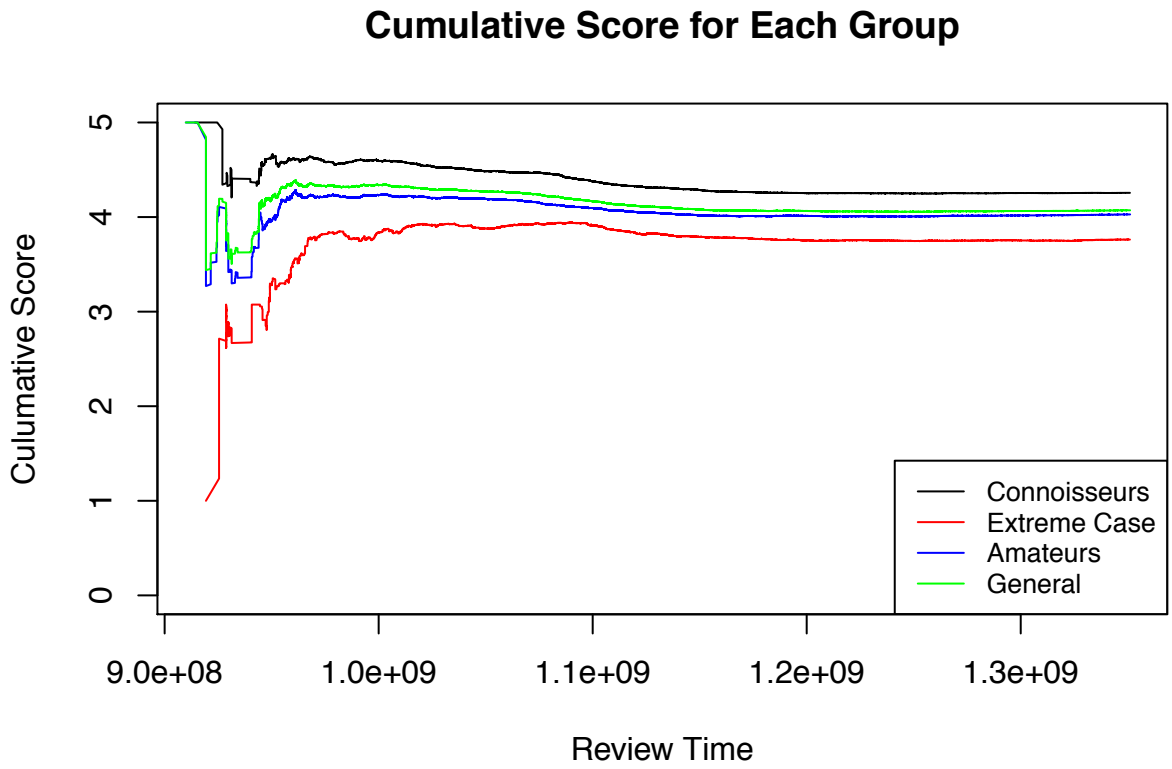



```
#par(mfrow=c(1,3))
#boxplot(connoi$word_ave,main='Connoisseurs',ylim=c(0,600))
#boxplot(word_for_user$word_ave,main='General',ylim=c(0,600))
#boxplot(ame$word_ave,main='Amateurs',ylim=c(0,600))
```

Compare of the Cumulative Score

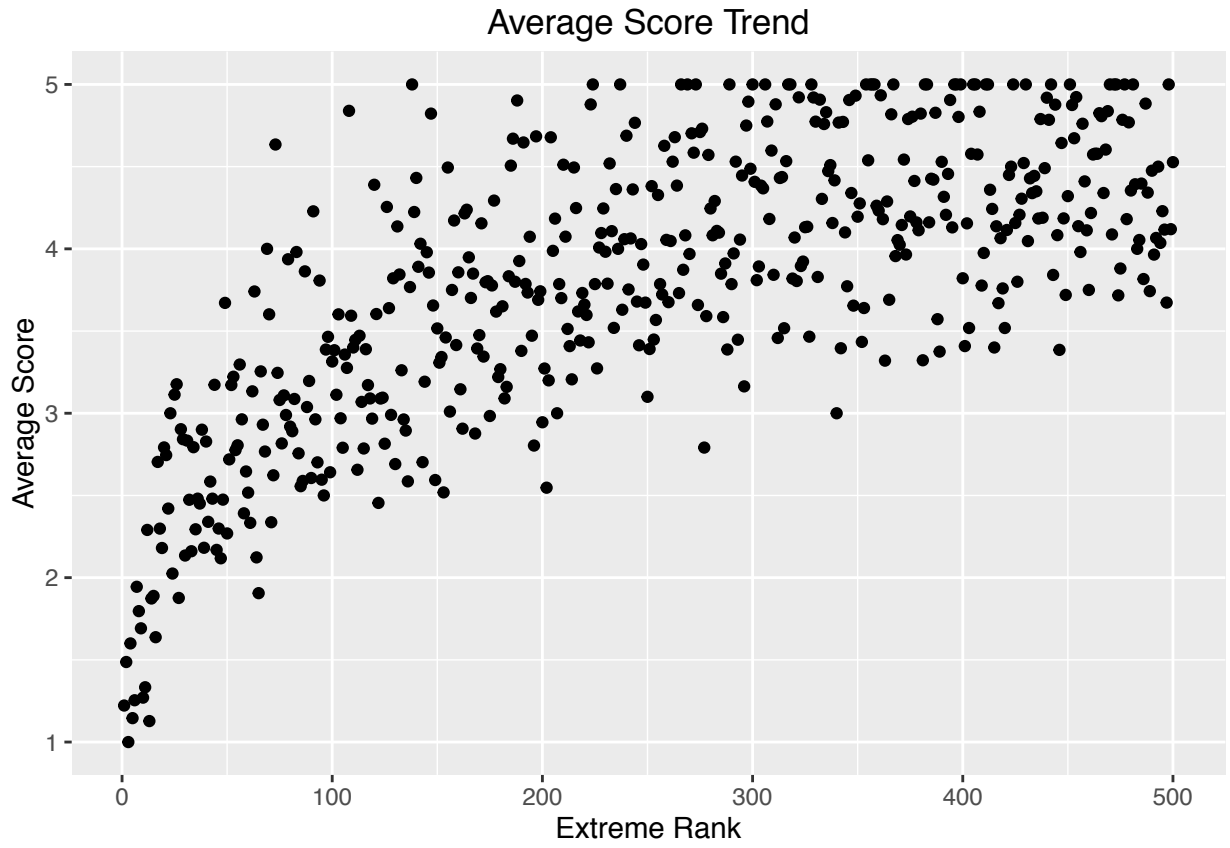
```
par(mfrow=c(1,1))
# calculate the cumulative score for Connoisseurs
t.c=left_join(data_new,connoi,by='review_userid')
t.c1=filter(t.c,word_ave!='NA')

t.c1=t.c1[order(t.c1$review_time),]
x.c=t.c1$review_time
y.c=cumsum(t.c1$review_score)/seq_along(t.c1$review_score)


# calculate the cumulative score for Extreme Cases
t.e=left_join(data_new,extreme,by='review_userid')
t.e1=filter(t.e,word_ave!='NA')

t.e1=t.e1[order(t.e1$review_time),]
x.e=t.e1$review_time
```

```
y.e=cumsum(t.e1$review_score)/seq_along(t.e1$review_score)


# calculate the cumulative score for Amateurs
t.a=left_join(data_new,ame,by='review_userid')
t.a1=filter(t.a,word_ave!='NA')

t.a1=t.a1[order(t.a1$review_time),]
x.a=t.a1$review_time
y.a=cumsum(t.a1$review_score)/seq_along(t.a1$review_score)

# calculate the cumulative score for General

t.g=data_new[order(data_new$review_time),]
x.g=t.g$review_time
y.g=cumsum(t.g$review_score)/seq_along(t.g$review_score)

plot(x.c,y.c,type='l',ylim=c(0,5),main='Cumulative Score for Each Group',xlab='Review Time',ylab = 'Cul
lines(x.e,y.e,col='red')
lines(x.a,y.a,col='blue')
lines(x.g,y.g,col='green')
legend('bottomright',legend=c('Connoisseurs','Extreme Case','Amateurs','General'),col=c('black','red','
```

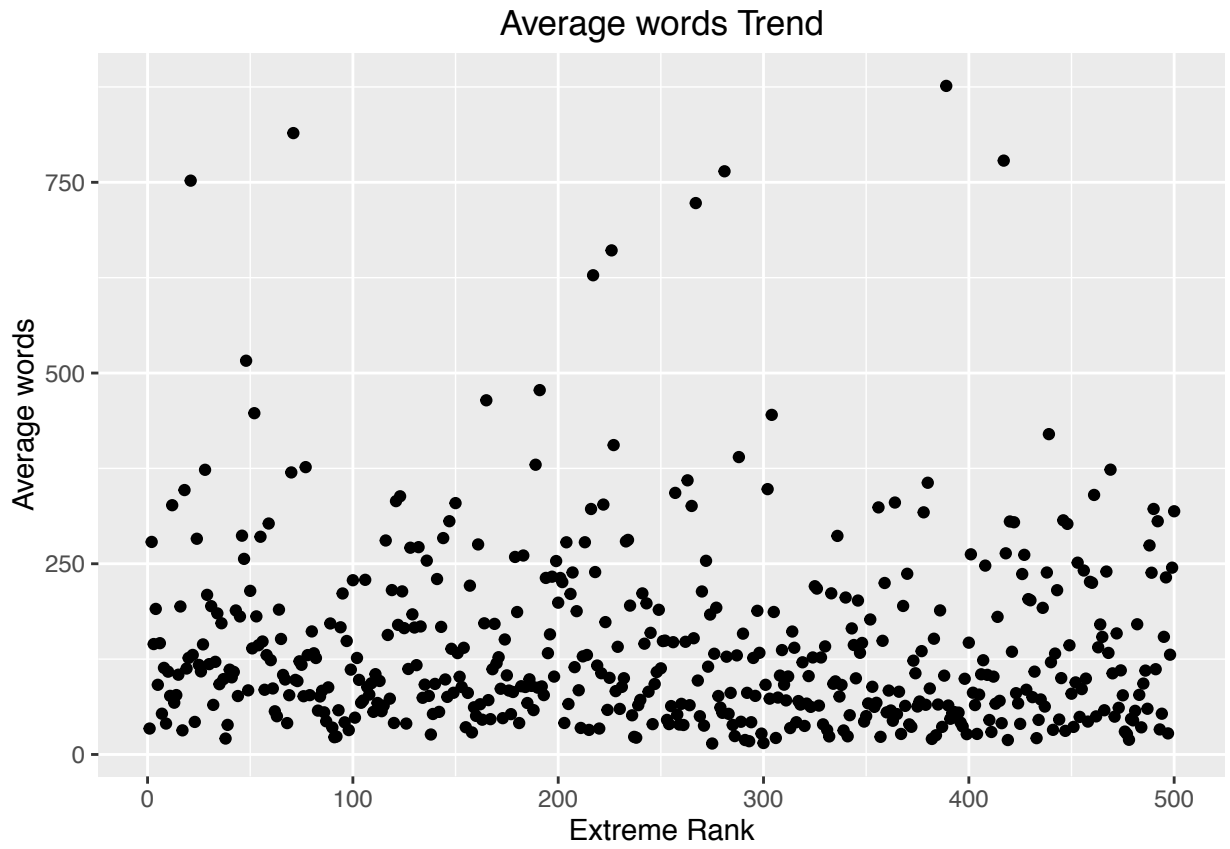



```
#boxplot(extreme$user.count,ylim=c(1,200),main='Number of Reviews')
#boxplot(extreme$UReview_ave,ylim=c(0,5),main='Average Score')
```

```
x=seq(1:500)
y=extreme$UReview_ave
ggplot()+geom_point(aes(x,y))+ggtitle('Average Score Trend')+xlab('Extreme Rank')+ylab('Average Score')
```




```
y2=extreme$word_ave
ggplot()+geom_point(aes(x,y2))+ggtitle('Average words Trend')+xlab('Extreme Rank')+ylab('Average words')
```



Analysis of Expert

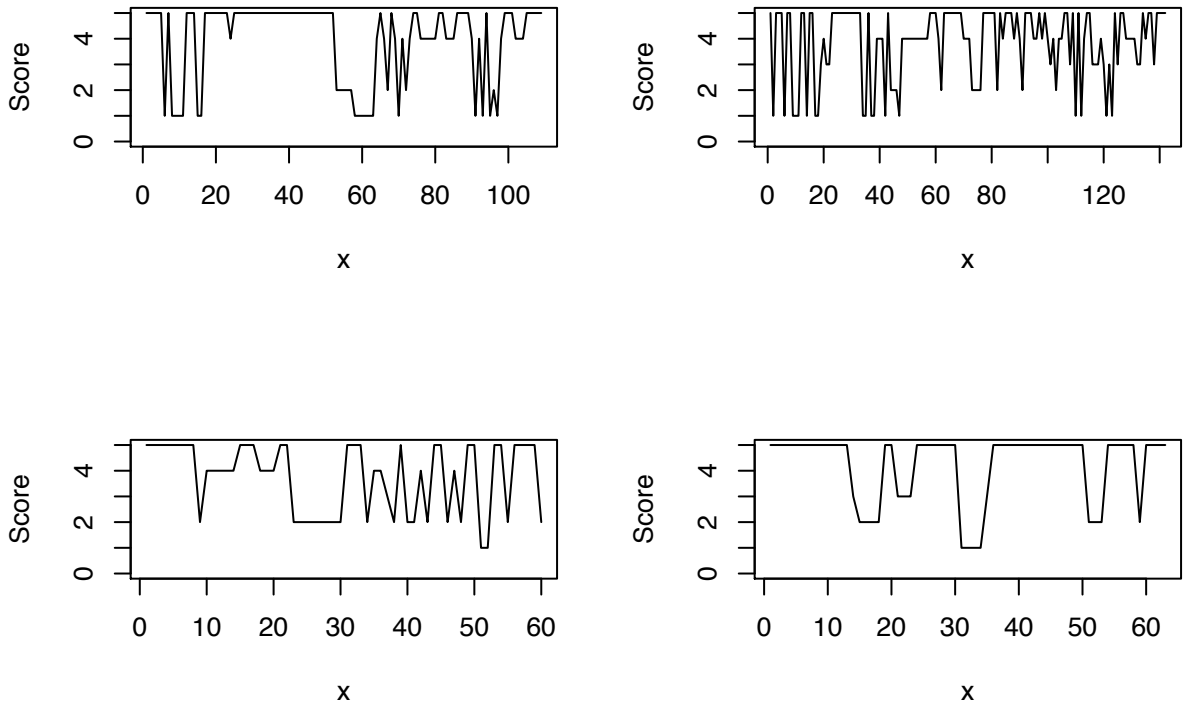
We also want to see the experts who have a good seperation of movies. Most of the experts have higher average score than the general. One reason for that is expert might more selective about movies that they only watch 'good' movies but this cause a problem of 'easy expert'. That is to say, it is easier to be an expert if you rate every movie high. In order to pick up experts who are critical, we calculate the sd of each individual expert of all the reviews they gave.

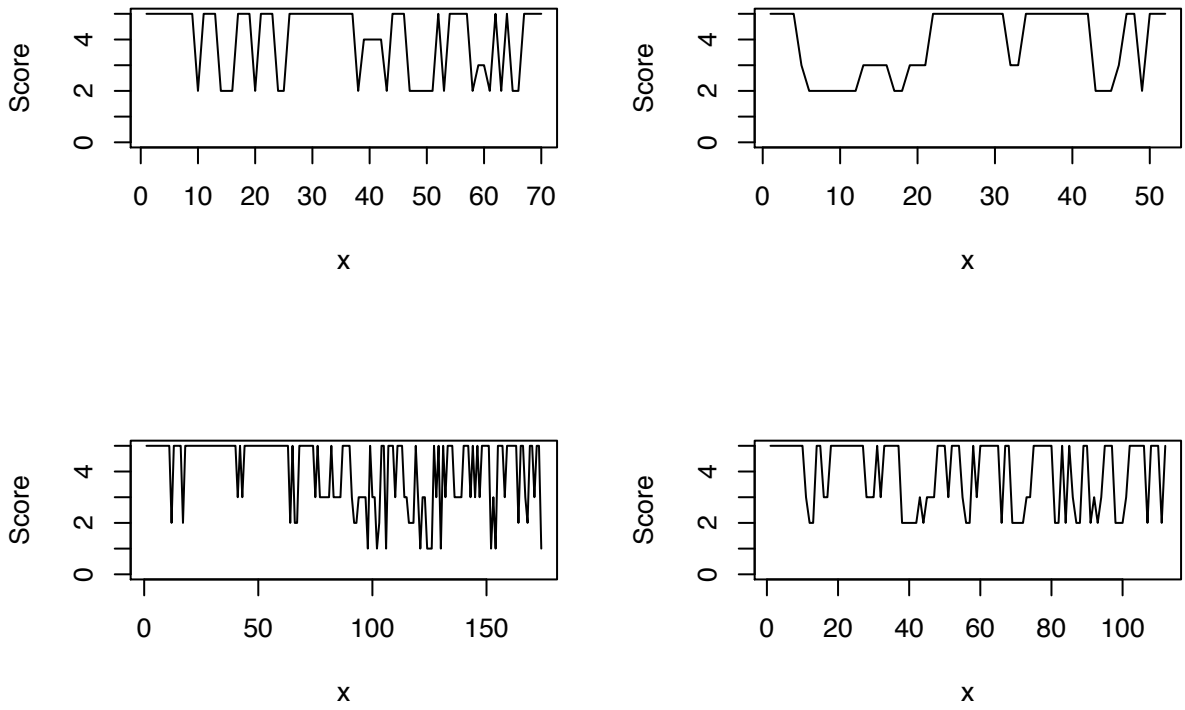
```
user.sd.new=data_new%>%
                group_by(review_userid)%>%
                summarize(
                  sd=sd(review_score,na.rm=T)
                )
connoi.new=left_join(connoi,user.sd.new,by='review_userid')
connoi.new=connoi.new[order(connoi.new$sd,decreasing = T),]
head(connoi.new)
```

```
## Source: local data frame [6 x 8]
##
##    review_userid user.count UReview_ave UReview_read UReview_help
##            (chr)      (int)       (dbl)        (dbl)        (dbl)
## 1 A37JKM7EFD0DIQ        109    3.935780     5.045872    0.6264788
## 2 A1OBPHRXHZF8P6        142    3.816901     6.260563    0.6069659
## 3 A298JV8C4ADLU7         60    3.733333     5.083333    0.6480769
```

```
## 4 A25QXIFEHR6900         63    4.206349    35.587302    0.7333848
## 5  AZ9JWGE1UGKZA         70    4.028571    15.371429    0.7577461
## 6 A38U7Z88Q1MDWL         52    3.826923     6.000000    0.7392872
## Variables not shown: sd_total (dbl), word_ave (dbl), sd (dbl)
```

```
critical=connoi.new[1:20,]
par(mfrow=c(2,2))
for (i in 1:8){
user=critical$review_userid[i]
user.temp=filter(data_new,review_userid==user)
x=seq(1:nrow(user.temp))
y=user.temp$review_score
plot(x,y,type='l',ylim=c(0,5),ylab='Score')
}
```





From the above plots we can see that 'critical' expert really have a clear seperation of scores. Some of the experts simply give 'good' movies 5 scores and 'bad' movies 1 score.

Sentimental Analysis

Yuhan Sun

April 11, 2016

Main reference: Sentimental Analysis in R Main Corpus: AFINN wordlist (http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010), Useful Adjectives for Describing Movies(http://member.tokoha-u.ac.jp/~dixonfdm/Writing%20Topics%20htm/Movie%20Review%20Folder/movie_descrip_vocab.htm)

Prepare the corpus

```
setwd('/Users/sunxiaohan/Desktop/project4/Sentiment')
afinn_list <- read.delim(file='AFINN/AFINN-111.txt', header=FALSE, stringsAsFactors=FALSE)
names(afinn_list) <- c('word', 'score')
afinn_list$word <- tolower(afinn_list$word)
#categorize words as very negative to very positive and add some movie-specific words
vNegTerms <- afinn_list$word[afinn_list$score==-5 | afinn_list$score==-4]

negTerms <- c(afinn_list$word[afinn_list$score==-3 | afinn_list$score==-2 | afinn_list$score==-1], "sec

posTerms <- c(afinn_list$word[afinn_list$score==3 | afinn_list$score==2 | afinn_list$score==1], "first-

vPosTerms <- c(afinn_list$word[afinn_list$score==5 | afinn_list$score==4], "uproarious", "riveting", "f
```

Sentiment Analysis function

```
library(plyr)
#function to calculate number of words in each category within a sentence
sentimentScore <- function(sentences, vNegTerms, negTerms, posTerms, vPosTerms){
  final_scores <- matrix('', 0, 5)
  scores <- laply(sentences, function(sentence, vNegTerms, negTerms, posTerms, vPosTerms){
    initial_sentence <- sentence
    #remove unnecessary characters and split up by word
    sentence <- gsub('[[:punct:]]', '', sentence)
    sentence <- gsub('[[:cntrl:]]', '', sentence)
    sentence <- gsub('\\d+', '', sentence)
    sentence <- tolower(sentence)
    wordList <- str_split(sentence, '\\s+')
    words <- unlist(wordList)
    #build vector with matches between sentence and each category
    vPosMatches <- match(words, vPosTerms)
    posMatches <- match(words, posTerms)
    vNegMatches <- match(words, vNegTerms)
    negMatches <- match(words, negTerms)
    #sum up number of words in each category
    vPosMatches <- sum(!is.na(vPosMatches))
```

```
    posMatches <- sum(!is.na(posMatches))
    vNegMatches <- sum(!is.na(vNegMatches))
    negMatches <- sum(!is.na(negMatches))
    score <- c(vNegMatches, negMatches, posMatches, vPosMatches)
    #add row to scores table
    newrow <- c(initial_sentence, score)
    final_scores <- rbind(final_scores, newrow)
    return(final_scores)
  }, vNegTerms, negTerms, posTerms, vPosTerms)
  return(scores)
 }
```

load the original text data

```
library(stringr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
load("/Users/sunxiaohan/Desktop/project4/connoi.Rdata")
data_use=readRDS('/Users/sunxiaohan/Desktop/project4/data_use.RDS')
load("/Users/sunxiaohan/Desktop/project4/extreme.Rdata")

# CON
Result.score.co=matrix(ncol=5,nrow=500)
colnames(Result.score.co) <- c('vNeg', 'neg', 'pos', 'vPos','Total')
for (i in 1:nrow(Result.score.co)){
  con.name=connoi$review_userid[i]
  text.raw=filter(data_use,review_userid==con.name)
  nr=nrow(text.raw)
  text.do=text.raw$review_text
  text=unlist(lapply(text.do, function(x) { str_split(x, "\n") }))
  Result <- as.data.frame(sentimentScore(text, vNegTerms, negTerms, posTerms, vPosTerms))
  Result1=Result[,2:5]
  for (a in 1:4){
```

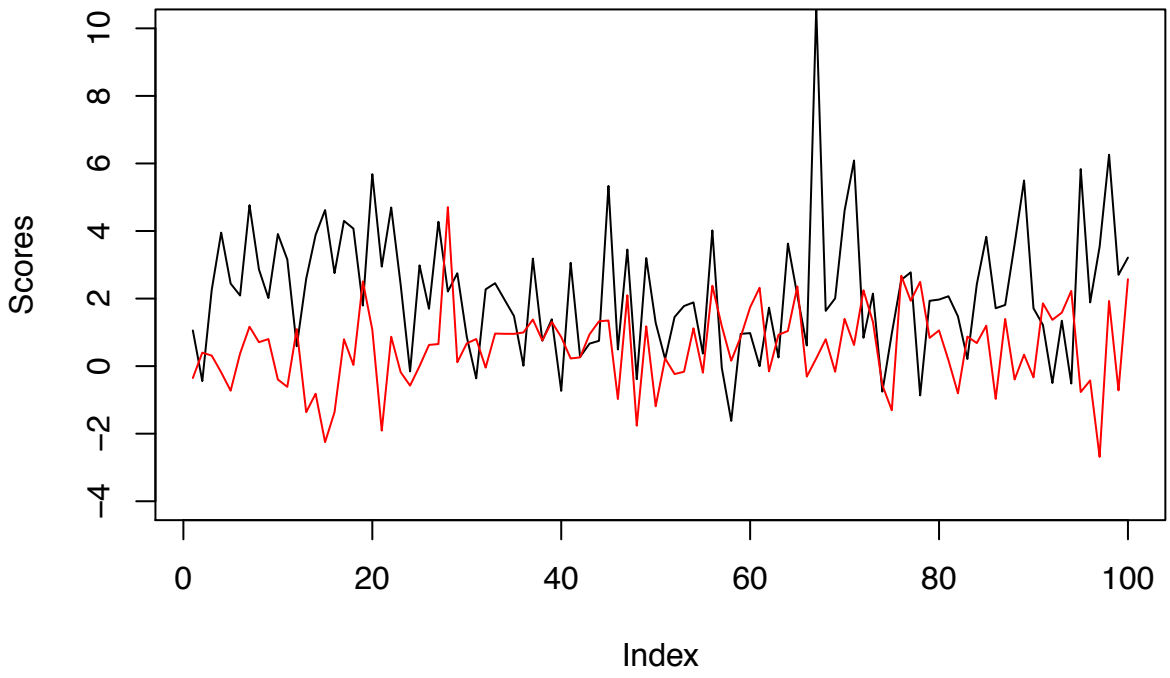
```
    Result.score.co[i,a]=sum(as.numeric(Result1[,a])-1)/nr
  }
  Result.score.co[i,5]=2*Result.score.co[i,4]+Result.score.co[i,3]-Result.score.co[i,2]-2*Result.score.
}
head(Result.score.co)
```

```
##            vNeg       neg      pos      vPos      Total
## [1,] 0.00000000 0.3943662 1.450704 0.0000000  1.0563380
## [2,] 0.33333333 0.9259259 1.148148 0.0000000 -0.4444444
## [3,] 0.00000000 1.2586207 1.724138 0.8965517  2.2586207
## [4,] 0.00000000 1.2619048 3.000000 1.1071429  3.9523810
## [5,] 0.00000000 0.2407407 1.240741 0.7222222  2.4444444
## [6,] 0.09090909 2.5363636 2.318182 1.2454545  2.0909091
```

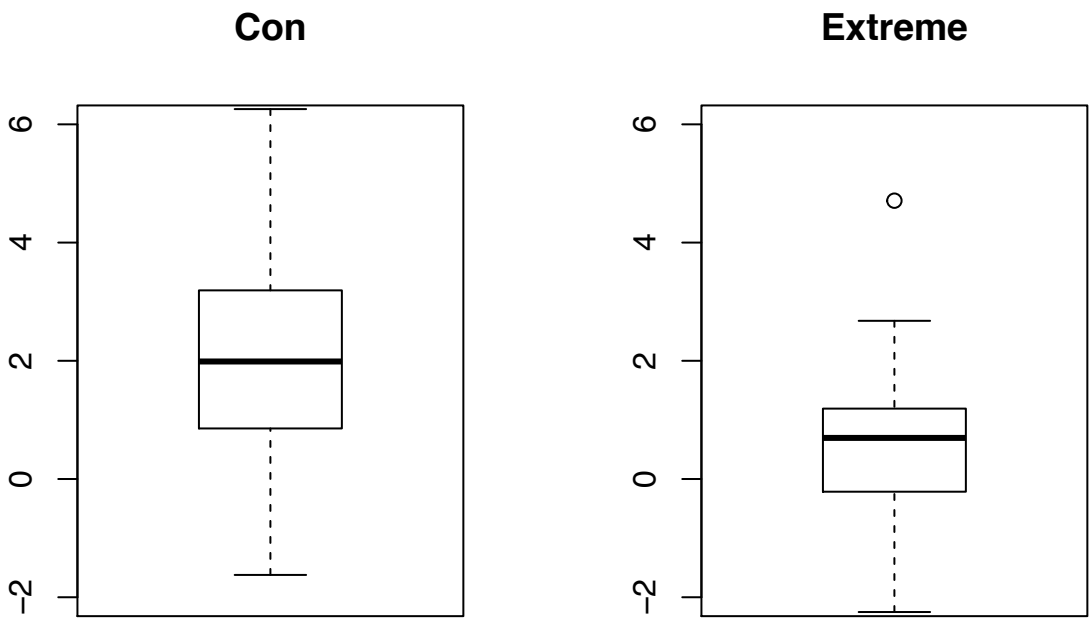
```
# Extreme Case
Result.score.ex=matrix(ncol=5,nrow=500)
colnames(Result.score.ex) <- c('vNeg', 'neg', 'pos', 'vPos','Total')
for (i in 1:nrow(Result.score.ex)){
  ex.name=extreme$review_userid[i]
  text.raw=filter(data_use,review_userid==ex.name)
  nr=nrow(text.raw)
  text.do=text.raw$review_text
  text=unlist(lapply(text.do, function(x) { str_split(x, "\n") }))
  Result <- as.data.frame(sentimentScore(text, vNegTerms, negTerms, posTerms, vPosTerms))
  Result1=Result[,2:5]
  for (a in 1:4){
    Result.score.ex[i,a]=sum(as.numeric(Result1[,a])-1)/nr
  }
  Result.score.ex[i,5]=2*Result.score.ex[i,4]+Result.score.ex[i,3]-Result.score.ex[i,2]-2*Result.score.
}
head(Result.score.ex)
```

```
##           vNeg      neg       pos       vPos      Total
## [1,] 0.0000000 1.333333 0.9444444 0.01851852 -0.3518519
## [2,] 0.0000000 5.469565 4.3043478 0.78260870  0.4000000
## [3,] 0.6229508 1.000000 1.4426230 0.55737705  0.3114754
## [4,] 0.1333333 2.800000 2.5166667 0.18333333 -0.1833333
## [5,] 0.0000000 1.472727 0.7454545 0.00000000 -0.7272727
## [6,] 0.0000000 1.661017 1.2881356 0.37288136  0.3728814
```

```
x1=Result.score.co[1:100,5]
plot(x1,type='l',ylim=c(-4,10),ylab='Scores')
x2=Result.score.ex[1:100,5]
lines(x2,col='red')
```




```
par(mfrow=c(1,2))
boxplot(x1,ylim=c(-2,6),main='Con')
boxplot(x2,ylim=c(-2,6),main='Extreme')
```


Con
Extreme
6
4
2
0
−2

Experts Recommendation

Jadie Zuo

April 11, 2016

Data Preparetion

We selecte the most deviated users: 100 most deviated reviewers, and 100 experts.

```
con <- load("connoi.RData")
ext <- load("extreme.RData")
mydata <- readRDS("users_50_products_100.RDS")
exp <- connoi[1:100,]
ext <- extreme[1:100,]
colnames(exp) <- c("userid","review_num","review_ave","help_num","help_score","dev")
colnames(ext) <- c("userid","review_num","review_ave","help_num","help_score","dev")
```

Prepare the node dataset: a 200 by 9 matrix with rows being the combination of the most deviated reviewrs and experts, and columns being the following features: * ID: identification number (a sequence from 1 to 200)
* userid: ID assigned by Amazon
* review_num: number of reviews created
* review_ave: average score of review
* help_num: number of helpfulness reviews by other users
* help_score: helpfulness score evaluated by other users
* dev: expertise measurement that is calculated by the deviation from his average review score to the overall review score
* type: binary variable with 1 being deviated reviewers and 2 being experts
* type.label: labels for type, extreme reviewers and experts

```
a <- load("node.RData")
node$type <- c(rep(1,100), rep(2,100))
node$type.label <- c(rep("Extreme Reviewers",100), rep("Experts",100))
node <- cbind(seq(1,200,1),node)
```

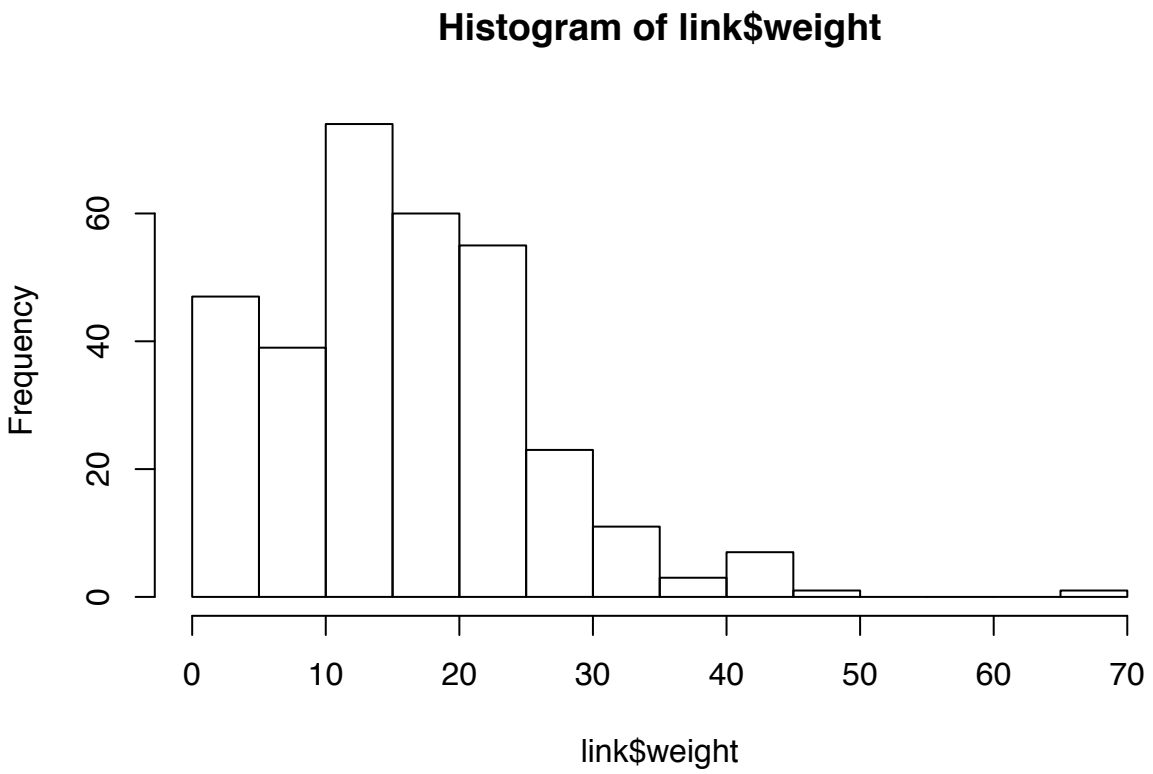
Prepare the edge dataset: a 321 by 4 matrix with rows being edges among 200 reviewers and following column factors:
* from: start point of an edge
* to: end point of an edge
* weight: number of movies that two nodes have commonly seen
* type: how strong the connection is, with 1 being the weight below 10 indicating a weak connection, 2 being the weight between 10 and 25 indicating a connection, 3 being weight above 25 indicating strong connection.

```
con <- matrix(nrow = 100,ncol = 100)
for (i in 1:100) {
  one <- mydata[which(mydata$review_userid == ext$review_userid[i]),]
  x1 = as.numeric(unique(one$product_productid))
  for (j in 1:100) {
    two <- mydata[which(mydata$review_userid == exp$review_userid[j]),]
    y1 = as.numeric(unique(two$product_productid))
    count <- length(intersect(x1, y1))
    con[i,j] <- count
```

```
  }
}

from <- c(1,1,1,1,1,2,2,2,2,3,3,3,3,4,5,6,6,6,6,6,6,6,6,7,7,9,9,9,9,9,9,9,9,
          9,10,10,10,10,10,10,10,11,11,11,11,11,11,11,13,13,13,13,13,13,14,
          14,14,15,16,16,16,17,17,18,18,19,19,19,20,20,21,21,21,21,22,26,26,
          26,26,26,26,27,27,28,28,28,29,29,29,29,29,29,30,32,32,32,34,34,35,
          35,35,36,36,37,37,38,38,38,38,38,38,38,38,39,39,39,39,41,41,42,42,
          42,42,42,42,42,42,43,43,43,43,43,44,44,45,45,45,45,46,46,46,46,46,
          46,46,46,46,47,47,47,49,49,49,49,49,49,50,50,50,50,50,50,50,50,50,
          50,50,50,51,51,52,52,52,54,54,54,54,54,54,54,54,55,55,56,56,56,56,
          57,57,57,57,58,58,58,59,60,60,60,61,62,63,64,64,64,64,64,66,67,67,
          68,68,69,70,70,70,70,71,71,72,72,72,72,72,72,72,72,72,73,73,73,73,
          73,74,74,76,76,76,77,77,77,78,78,78,78,79,79,80,80,80,80,81,81,81,
          81,82,82,82,82,83,83,84,84,84,84,84,84,85,85,85,86,86,87,87,87,87,
          88,89,89,89,89,89,89,90,90,91,91,91,91,92,92,92,92,93,94,94,95,96,
          96,96,96,96,96,97,98,98,98,98,98,98,98,99,99,99,99,99,99,99,100,
          100,100,100)
to <- c(34,64,82,83,95,67,75,79,83,15,35,72,94,52,92,27,30,43,59,72,6,14,57,
        11,93,22,32,54,92,94,75,76,83,67,29,67,43,64,92,83,95,83,92,70,73,86,
        87,63,90,43,92,67,75,64,99,90,93,75,50,32,75,90,93,68,61,83,91,92,64,
        17,12,92,96,90,63,40,92,61,24,54,83,70,90,66,83,92,8,14,43,59,83,100,
        29,68,90,97,83,91,83,56,43,90,75,47,83,1,32,54,67,94,90,98,92,83,92,
        75,63,21,92,55,68,82,28,75,90,22,99,16,43,64,92,100,92,67,90,75,92,42,
        93,44,58,63,39,75,89,32,92,35,92,4,38,59,60,83,20,92,95,65,99,62,10,
        67,73,56,75,92,91,90,83,53,90,83,91,53,59,43,83,88,56,75,92,48,92,56,
        75,19,63,48,49,75,83,91,90,93,85,9,26,83,65,83,99,82,43,21,75,90,91,49,
        64,83,100,83,92,67,53,75,83,90,83,8,19,42,43,64,91,92,100,68,62,6,14,
        57,47,83,90,92,32,98,82,53,91,90,83,82,83,90,41,66,32,83,53,93,90,85,
        78,62,83,55,92,90,70,12,44,75,92,45,35,72,94,83,55,68,83,24,54,53,48,
        41,75,43,19,63,56,50,39,40,43,92,92,49,75,83,79,82,92,96,73,59,69,56,
        75,92,75,64,4,63,70,73,86,87,85,90,83,34,44,92,95,11,1,42,92)
weight <- c()
for (i in 1:321){
  weight[i] <- con[from[i],to[i]]
}
```

Take a look at how "weight" is distributed:

```
hist(link$weight)
```



Creat a categorical varaible using the weight variable to describe the level of similarity between reviewers. According to the histogram, use 10 and 25 as two cut off points:

```
type <- c()
for (i in 1:321){
  if (weight[i] < 10) {
    temp <- 1
  }
  else if (weight[i] < 25) {
    temp <- 2
  }
  else {
    temp <- 3
  }
  type[i] <- temp
}
link <- data.frame(from,to,weight,type)
colnames(link) <- c("from", "to", "weight", "type")
rownames(link) <- NULL
```

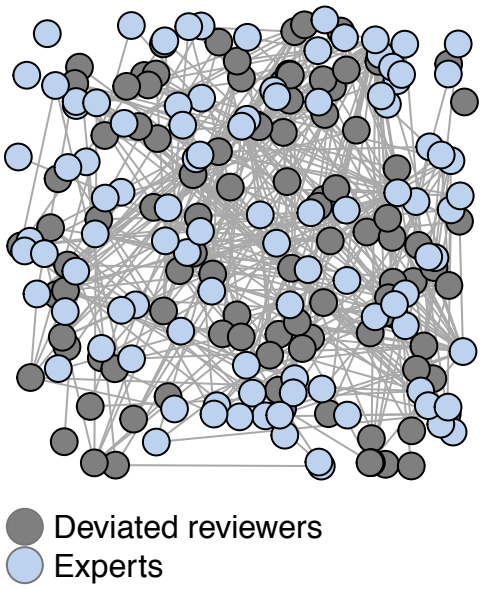
Network Plots

Network layout using igraph:

```
library(igraph)
library(RColorBrewer)
net <- graph.data.frame(link, node, directed=T)
net <- simplify(net, remove.multiple = F, remove.loops = T)
colrs <- c("gray50", "lightsteelblue2")
```

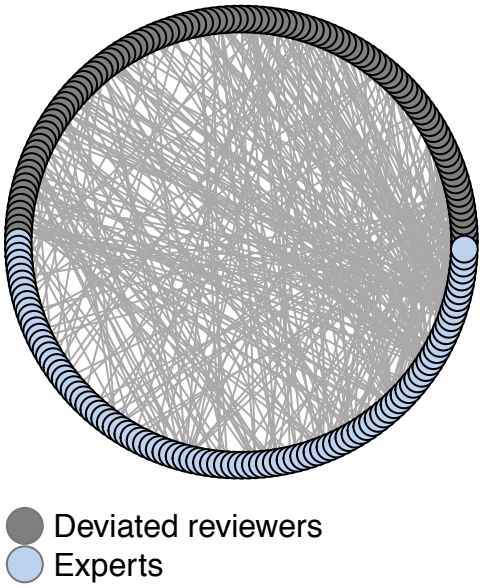
Random Network Layout

```
plot(net, vertex.size=12, edge.arrow.size=0, edge.curved=0,vertex.color=colrs[V(net)$type],
     vertex.frame.color="black",vertex.label=NA, layout=layout.random)
legend(x=-1.1, y=-1.1, c("Deviated reviewers","Experts"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2.5, bty="n", ncol=1)
```



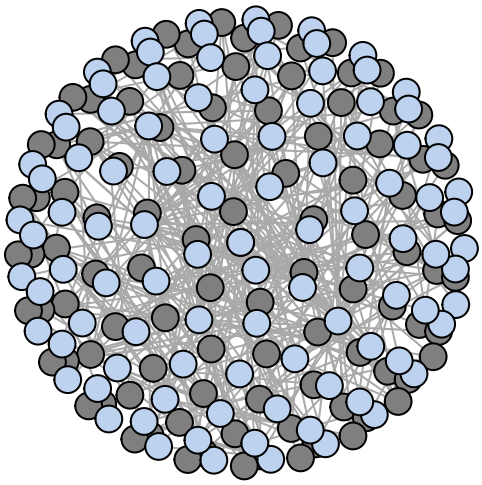
Circle Layout

```
plot(net,vertex.size=12, edge.arrow.size=0, edge.curved=0,vertex.color=colrs[V(net)$type],
     vertex.frame.color="black",vertex.label=NA,layout=layout.circle(net))
legend(x=-1.1, y=-1.1, c("Deviated reviewers","Experts"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2.5, bty="n", ncol=1)
```



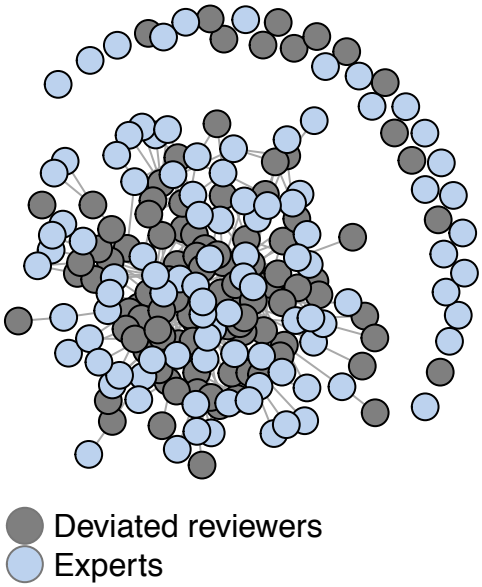
3D sphere layout:

```
plot(net,vertex.size=12, edge.arrow.size=0, edge.curved=0,vertex.color=colrs[V(net)$type],
     vertex.frame.color="black",vertex.label=NA,layout=layout.sphere(net))
legend(x=-1.1, y=-1.1, c("Deviated reviewers","Experts"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2.5, bty="n", ncol=1)
```



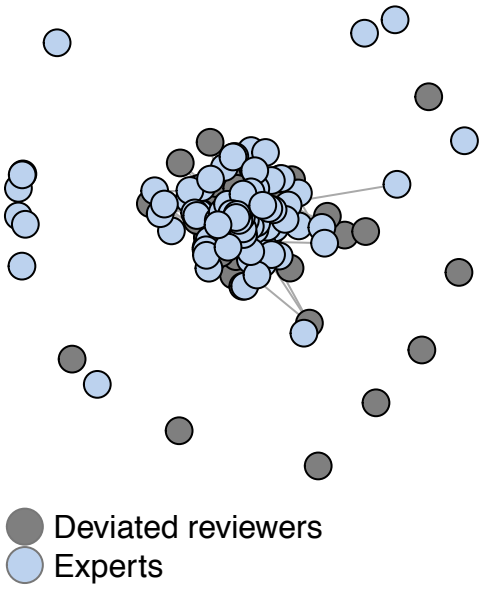
The Fruchterman-Reingold force-directed algorithm:

```
plot(net,vertex.size=12, edge.arrow.size=0, edge.curved=0,vertex.color=colrs[V(net)$type],
     vertex.frame.color="black",vertex.label=NA,layout=layout.fruchterman.reingold)
legend(x=-1.1, y=-1.1, c("Deviated reviewers","Experts"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2.5, bty="n", ncol=1)
```



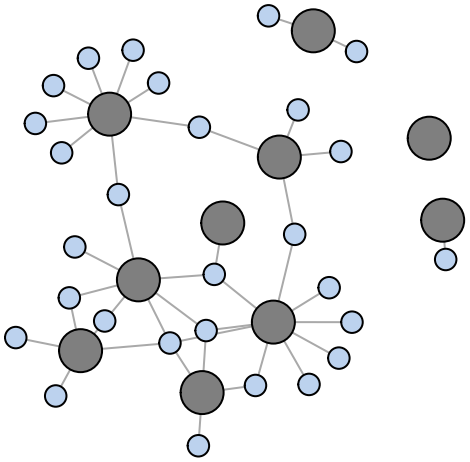
The Kamada Kawai forced-directed algorithm:

```
plot(net,vertex.size=12, edge.arrow.size=0, edge.curved=0,vertex.color=colrs[V(net)$type],
     vertex.frame.color="black",vertex.label=NA,layout=layout.kamada.kawai(net))
legend(x=-1.1, y=-1.1, c("Deviated reviewers","Experts"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2.5, bty="n", ncol=1)
```

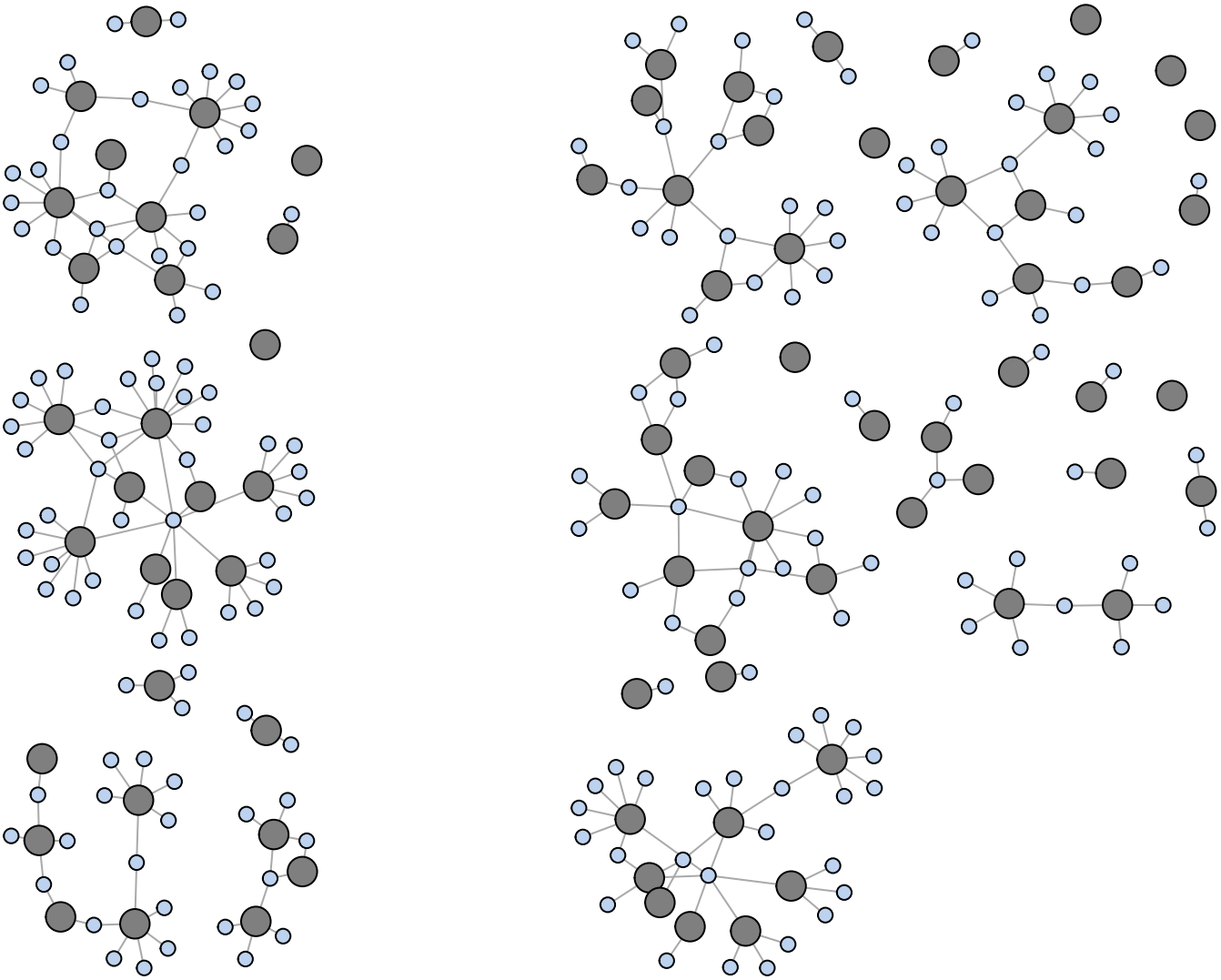


Connect experts with the needed (10 deviated reviewers)

```
colrs <- c("gray50", "lightsteelblue2")
node.new <- node[c(1:10,101:200),]
link.new <- link[which(link$from < 11),]
node.new <- node[c(1:10,unique(link.new$to)),]
net.new <- graph.data.frame(link.new, node.new, directed=T)
net.new <- simplify(net.new, remove.multiple = F, remove.loops = T)
l <- layout.fruchterman.reingold(net.new, repulserad=vcount(net.new)^3,
                                 area=vcount(net.new)^2.4)
plot(net.new, vertex.size=20/V(net.new)$type, edge.arrow.size=0, edge.curved=0,
     vertex.color=colrs[V(net.new)$type], vertex.frame.color="black",
     vertex.label=NA, layout=l)
```

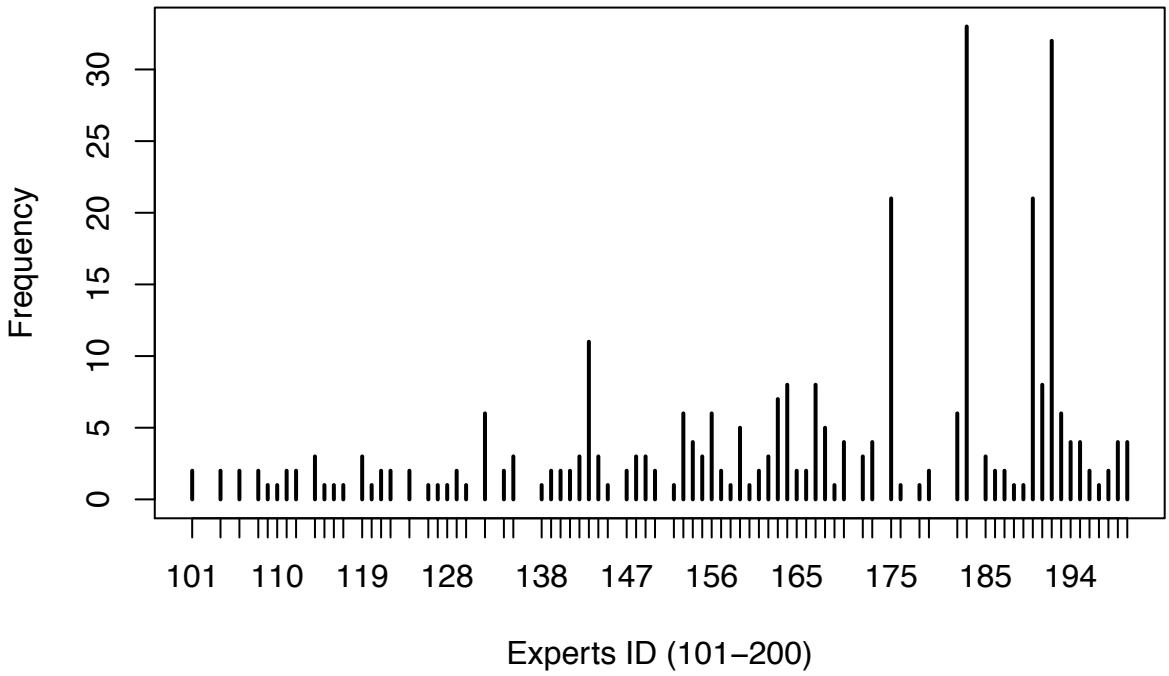


Expert recommendation for all the deviated users:



Exam the involvement of experts in the system:

```
plot(table(link$to),xlab="Experts ID (101-200)", ylab="Frequency")
```




```
length(order(table(link$to)))
```

```
## [1] 80
```

```
#There are 80 experts out of 100 recommended to the deviated reviewers.
#Print 10 most advanced expters
table(link$to)[order(table(link$to))[71:80]]
```

```
##
## 193 163 164 167 191 143 175 190 192 183
##   6   7   8   8   8  11  21  21  32  33
```

Who are they?

```
adv <- exp[c(93,63,64,67,91,43,75,90,92,83),c(2:6)]
adv
```

```
##    review_num review_ave  help_num help_score       dev
## 93         96   4.416667  7.375000  0.7293581 0.3307292
## 63        223   4.654709 17.699552  0.7539035 0.2780722
## 64         70   4.371429 14.600000  0.7800065 0.2793805
## 67        238   4.689076 35.180672  0.8531945 0.2890479
## 91         64   4.609375 12.078125  0.8715278 0.3287300
## 43         54   4.333333 11.333333  0.9220779 0.2384003
## 75         64   4.718750 10.031250  0.7560153 0.3021759
## 90         59   4.440678 28.474576  0.7509383 0.3286450
## 92        406   4.349754 20.460591  0.7909175 0.3297312
## 83         76   4.197368  7.855263  0.8715479 0.3152513
```

```r
colMeans(adv)
```

```
##  review_num  review_ave    help_num  help_score         dev
## 135.0000000   4.4781138  16.5088363   0.8079487   0.3020163
```

```r
exp_sub <- exp[,c(2:6)]
colMeans(exp_sub)
```

```
##  review_num  review_ave    help_num  help_score         dev
## 125.9300000   4.5914989  15.6489400   0.8000546   0.2402854
```

1. Average number of reviews for movies is considerable high than the experts popylation
   —> No surprise

2. Average review scores for the 10 advanced experts is lower than the experts population
   —> More critical?

3. Deviation of the 10 advanced experts is higher than the experts population
   —> Professional perspective?

Recommendation Algorithms and Evaluation

Group8

April 12, 2016

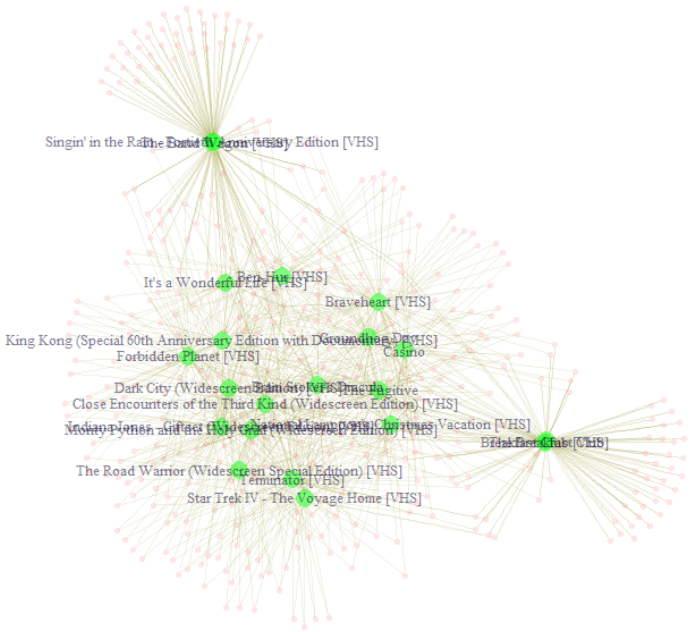
- Social Network Analysis & Clustering
- Biclustering Analysis
- Chi-Square Analysis
- Model Evaluation

1 Social Network Analysis & Cluster

In order to better understand movie-user biparite network, the following parts fisrt visualize a sample biparite network and show some primary characteristics. Then by converting the biparite network to one-mode movie network, we can have a deeper understanding on movie grouping. Furthermore, based on the idea of movie grouping, we introduced fast greedy algorithm to deal with our large-scale network. And finally we are able to reach our movie recommendation system.

1.1 Biparite Network:

```
whole_data <- readRDS('data_use.RDS')
df1 <- data.frame( user=whole_data[1:2000,]$review_userid, movie=whole_data[1:2000,]$
product_name, stringsAsFactors = F)
m1 <- table( df1 )
user_movie1 <- as.matrix( m1 )

oldw <- getOption("warn")
options(warn = -1)

i2mode<-graph.incidence(user_movie1, mode=c('all'))
V(i2mode)$color[1:1150] <- rgb(1,0,0,.5)
V(i2mode)$color[1151:1171] <- rgb(0,1,0,.5)

V(i2mode)$label <- V(i2mode)$name
V(i2mode)$label.color <- rgb(0,0,.2,.5)
V(i2mode)$label.cex <- .4
V(i2mode)$size <- 6
V(i2mode)$frame.color <- NA

E(i2mode)$color <- rgb(.5,.5,0,0.2)
E(i2mode)$width <- 0.5

# simplied 2 mode graph (degree > 1)
i2mode <- delete.vertices(i2mode, V(i2mode)[degree(i2mode)==1])

V(i2mode)$label[1:427] <- NA
V(i2mode)$color[1:427] <-  rgb(1,0,0,.1)
V(i2mode)$size[1:427] <- 2

E(i2mode)$width <- .3
E(i2mode)$color <- rgb(.5,.5,0,.1)

options(warn = oldw)

plot(i2mode, layout=layout.fruchterman.reingold)
```



This is a small sample biparite network. Green node means moive. Red node means reviewer. Edge means there is a review from a reviewer to a movie. The position can show the closeness of movies in some sense.

1.2 One-Mode Movie Network

```
##########movie overlap

gmovie_mode1 <- tcrossprod(t(user_movie1))
olmovie <- gmovie_mode1/diag(gmovie_mode1)
olmovie[is.na(olmovie)] <- 0

magdiag <-diag(olmovie)
magallg <- graph.adjacency(olmovie, weighted=T)

# Degree
V(magallg)$degree <- degree(magallg)

# Betweenness centrality
V(magallg)$btwcnt <- betweenness(magallg)


magall<-olmovie

magallgt1 <- magall
magallgt1[magallgt1<0.2] <- 0
magallggt1 <- graph.adjacency(magallgt1, weighted=T)

# Removes loops:
magallggt1 <- simplify(magallggt1, remove.multiple=FALSE, remove.loops=TRUE)

magallggt1$layout <- layout.fruchterman.reingold(magallggt1)
V(magallggt1)$label <- V(magallggt1)$name



# Set vertex attributes
V(magallggt1)$label <- V(magallggt1)$name
V(magallggt1)$label.color <- rgb(0,0,.2,.6)
V(magallggt1)$size <- 6
V(magallggt1)$frame.color <- NA
V(magallggt1)$color <- rgb(0,0,1,.5)

# Set edge attributes
E(magallggt1)$arrow.size <- .3

# Set edge gamma according to edge weight
egam <- (E(magallggt1)$weight+.1)/max(E(magallggt1)$weight+.1)
E(magallggt1)$color <- rgb(.5,.5,0,egam)


plot(magallggt1)
```

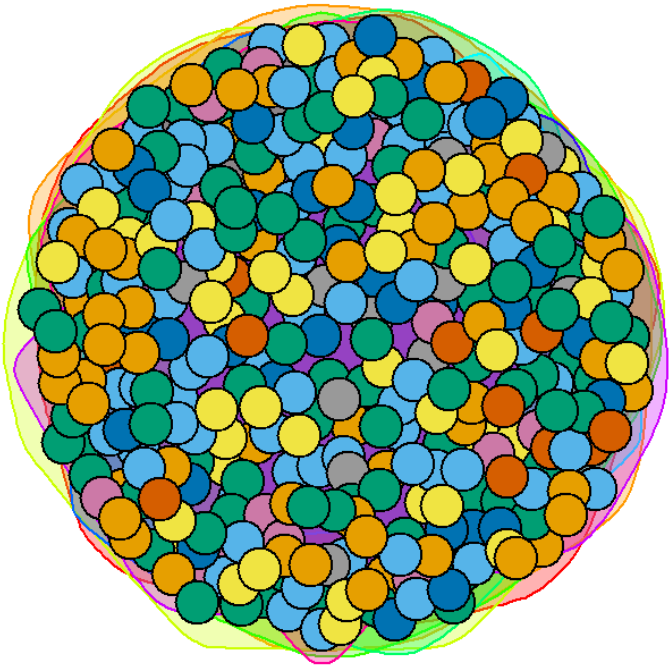
This network is One-Mode Movie Network, which also shows a strong overlap of reivewers between two movies. If there is an arrow from movie A to movie B, at least 20% of reviewers of movie A also have given reviews to movie B. This phenomenon can indicate that we can recommend movie B to the audience of movie A. This method is meaningful but pretty slow. Thus we need to apply other methods with the similar idea.

1.3 Community Dection:

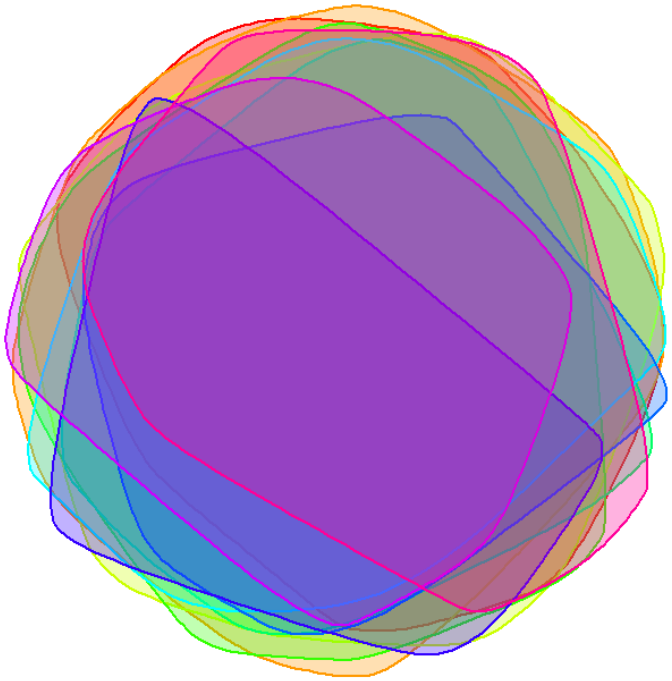
Faster Greedy Algorithm tries to find dense subgraph, also called communities in graphs via directly optimizing a modularity score. Modularity designed to measure the strength of division of a network into modules. Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.

For one-mode movie network, nodes represent movies. Edge means that there is at least one common reviewer for the two movies connected. The weight of an edge means the number of common reviewers for two movies connected. Therefore, a community in network means movies receiving attention from a closed group of people, which shows the closeness of movies.

Faster Greedy Node: Node represents movie; Color of node represents community.



Faster Greedy Group: Color of area means cluster. Below is the size of each cluster



```
## Community sizes
##  1  2  3  4  5  6  7  8  9 10
## 45 67 85 44 32 13 12 11 15 10
```

1.4 Recomendation Algorithm 1:

This recommendation algorithm based on community detection of movie network and ranking of movie score. Intuitively, the result will be a good movie fitting people with similar taste.

Advantage: 1. considered attention of movies from people with similar taste 2. considered quality of movies 3. fast

Disadvantage: 1. didn't consider past watching experience of the user 2. didn't include randomness into recommendation

```
# input
movie_name='American Werewolf in London [VHS]'
```

```
for (i in 1:length(fc)) {
  idx=match(movie_name,unlist(fc[i]))
  if(!is.na(idx)){
    break
  }
}
recommend_list=unlist(fc[i])
recommend_list=recommend_list[recommend_list!=movie_name]

score=rep(0,length(recommend_list))

for (i in 1:length(recommend_list)) {
  if(substr(movie_name,1,4)==substr(recommend_list[i],1,4)){
    recommend_list=recommend_list[recommend_list!=recommend_list[i]]
  }
  else{score[i]=mean(subset(movie_user_data,recommend_list[i]==movie_user_data$produc
t_name)$review_score)}

}

op=data.frame(cbind(recommend_list,score))
op<-op[order(op$score,decreasing=TRUE),]
op[1:3,]
```

```
##                                                 recommend_list
## 102            Jaws - 25th Anniversary Collector's Edition [VHS]
## 103 Jaws (25th Anniversary Widescreen Collector's Edition) - DTS
## 108            Jaws (Widescreen Anniversary Collector's Edition)
##                score
## 102 4.86206896551724
## 103 4.83908045977012
## 108 4.82758620689655
```

Output: top3 movies and their corresponding scores

2 Biclustering

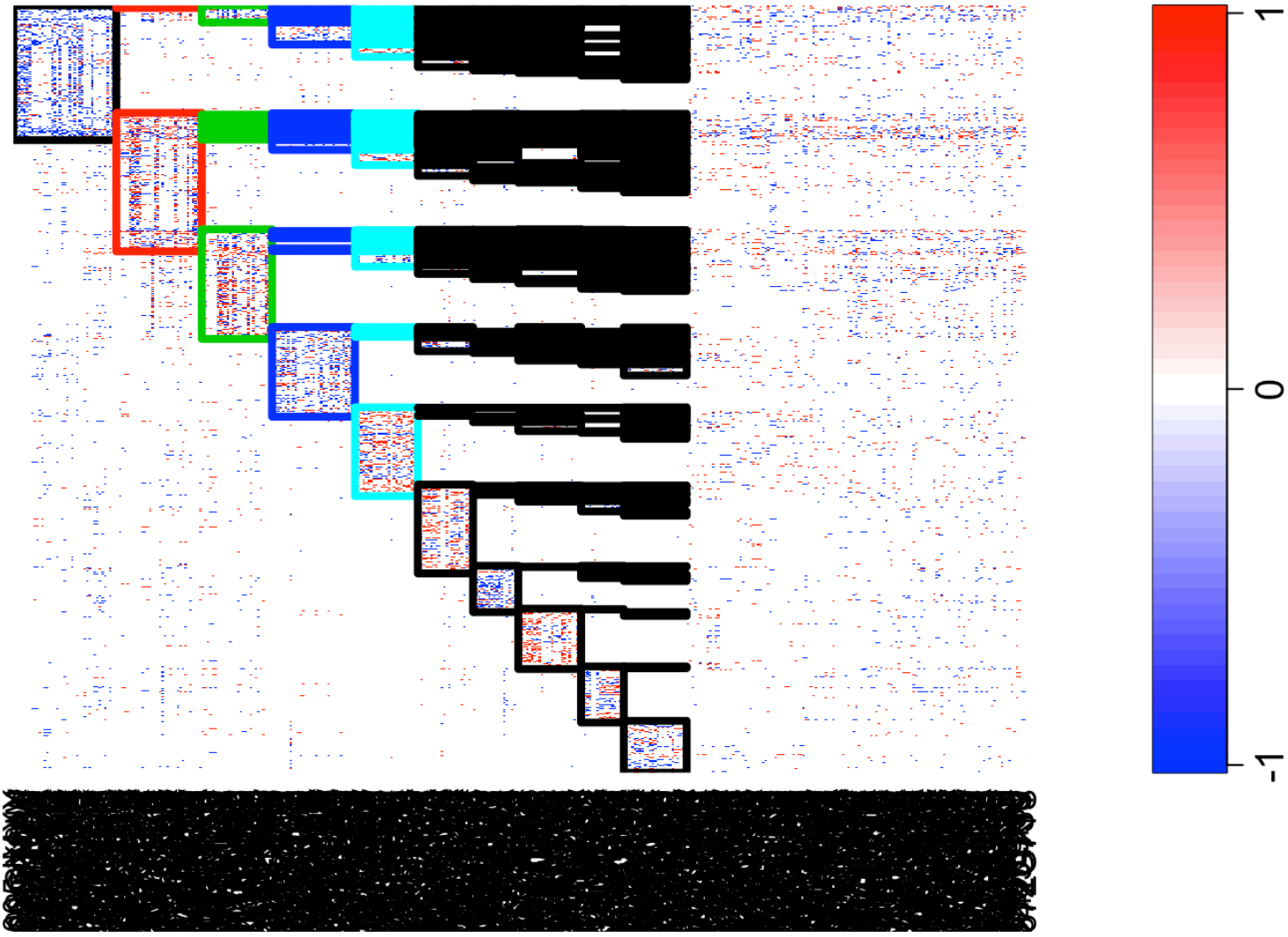
Biclustering is an important new technique in two way data analysis. With biclustering, we can do simultaneous clustering of 2 dimensions; Large datasets where clustering leads to diffuse results; Only parts of the data influence each other.

2.1 Biclustering based on user id and product id:

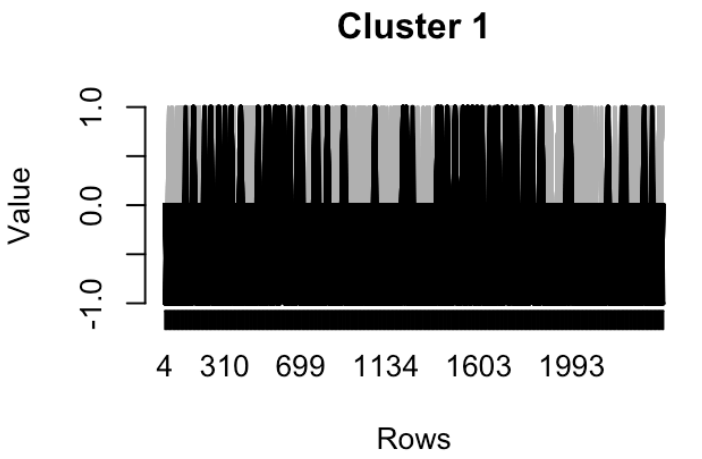
First of all, we do biclustering based on user id and product id. Due to concerns about memory and running time, we extract a subset from the original dataset. For movies (403), those that are reviewed by experts and have more than 35 reviews are selected. For user id (2486), only reviewers who have watched the above movies are chosen.

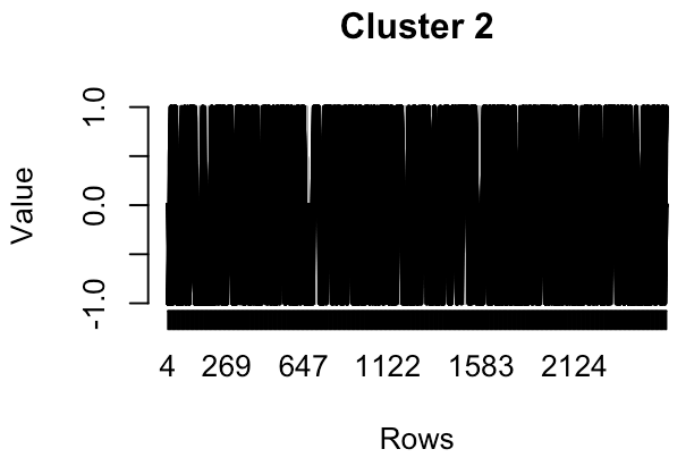
To creating the sparse matrix, user id represents rows, and product id represents columns. If user i gives review score 5 to product j, then the entry (i,j) would be 1, if user i gives review score below 5 to product j, then the entry (i,j) would be -1, if user i hasn't given a review to product j, then the entry (i,j) would be 0.

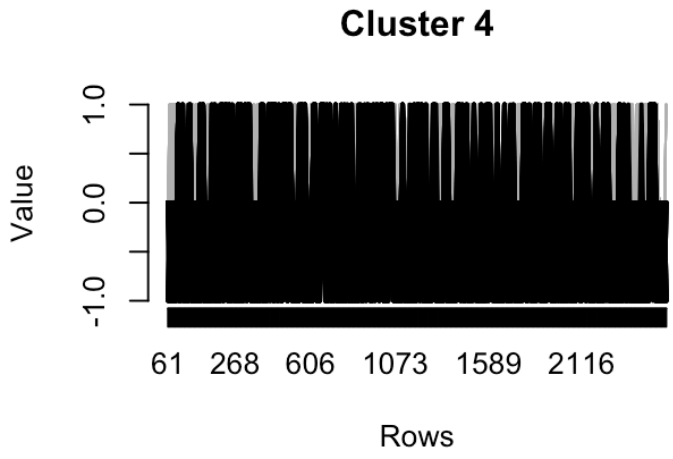
```
## Bicluster 1........
## Bicluster 2..........
## Bicluster 3........
## Bicluster 4.....................................
## Bicluster 5.......................
## Bicluster 6.............
## Bicluster 7.................
## Bicluster 8.............
## Bicluster 9..................
## Bicluster 10............
```



We also can use parallelCoordinates plot to see every cluster's characteristics. Just show first 4 as examples.







So far, we can give a recommendation system according to this approach. For example, if user “A17IW44FV0HUTY” wants to find some recommendations, we will firstly locate his/her cluster, second we select all movies that are in this cluster and haven’t been watched by user “A17IW44FV0HUTY”, and then we recommend top n (n is a customized number of recommendations, for instance, n=3) movies based on average review socre.

Finally, we have the results.

```
## [1] "Jaws"
## [2] "United 93 [DVD + Digital Copy] (Universal's 100th Anniversary)"
## [3] "Million Dollar Baby"
```

On the other hand, if one user likes movie “B00000I4XR”, which is “Jaws”, again, we will firstly locate its cluster, second we select all other movies that are in this cluster, and then we recommend top n (n is a customized number of recommendations, for instance, n=3) movies based on average review socre.

We now have the results.

```
recommend_new_final[1:3]
```

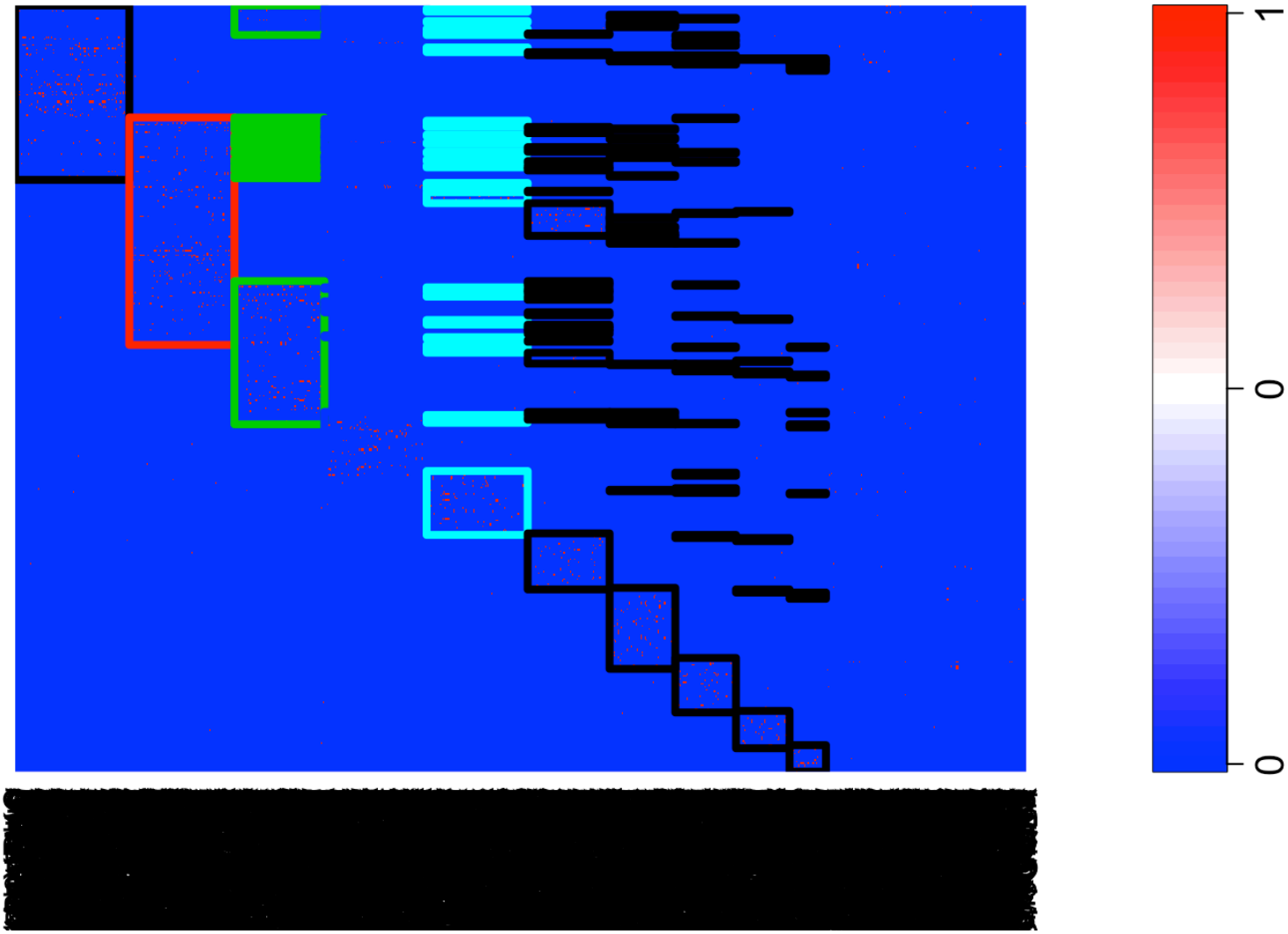
```
## [1] "United 93 [DVD + Digital Copy] (Universal's 100th Anniversary)"
## [2] "Master and Commander - The Far Side of the World [VHS]"
## [3] "Pan's Labyrinth (New Line Two-Disc Platinum Series)"
```

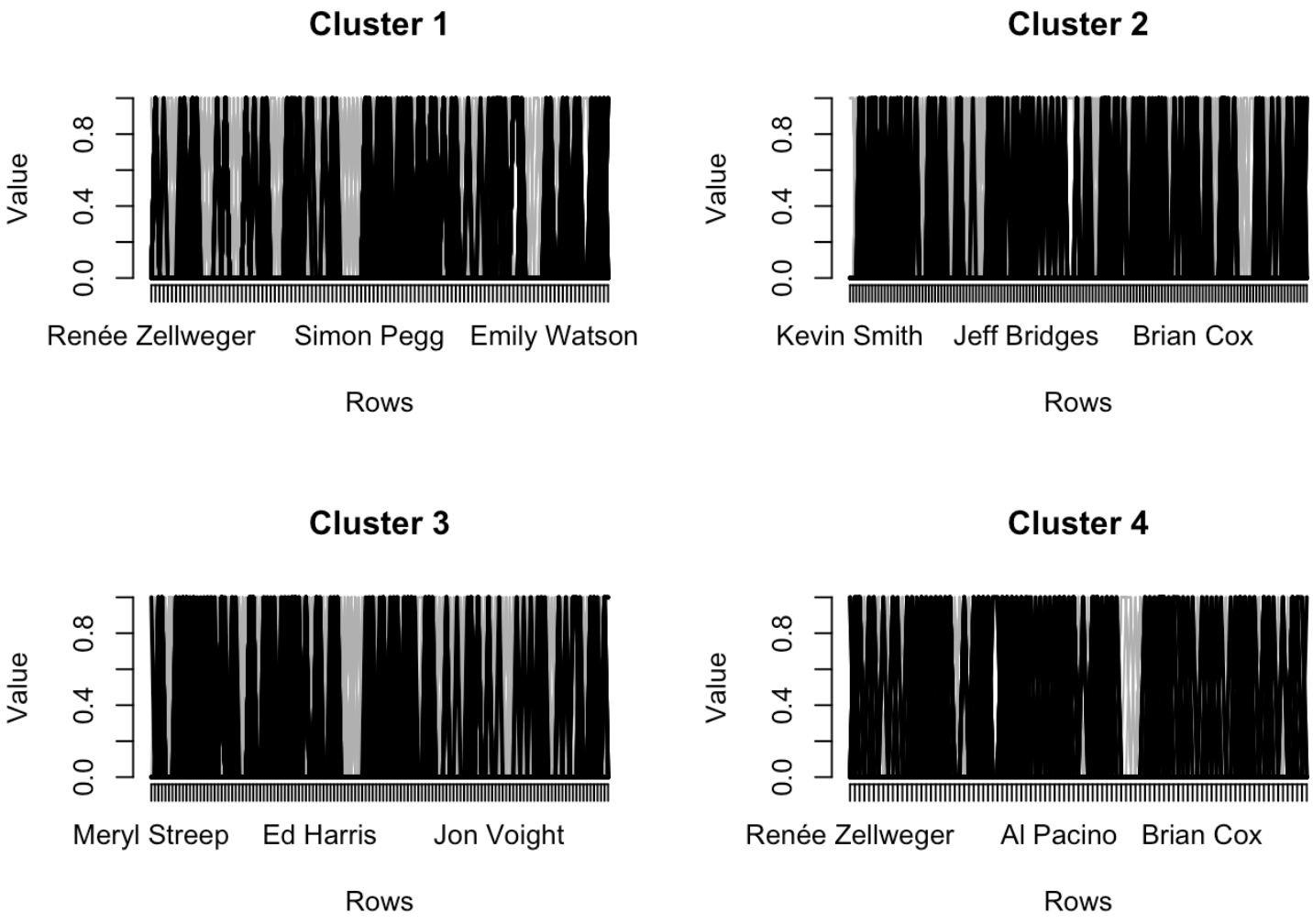
2.2 Biclustering based on actors' and directors' names and product id:

Second, we do biclustering based on actors and directors and product id. Also, since the dataset is too large, we extract a subset with last 1000 movies and first 1000 actors from the original dataset.

To creating the sparse matrix, actors' names represent rows, and product id represents columns. If actor i performed in movie j, then the entry (i,j) would be 1, otherwise would be 0.

```
## Bicluster 1....
## Bicluster 2..............
## Bicluster 3.....
## Bicluster 4....
## Bicluster 5....
## Bicluster 6....
## Bicluster 7....
## Bicluster 8....
## Bicluster 9....
## Bicluster 10....
```


1
0
0



So far, we can give a recommendation system according to this approach. For example, if one user inputs a movie that he/she likes. Again, we will firstly locate its cluster, second we select all movies that are in this cluster, and then we recommend top n (n is a customized number of recommendations, for instance, n=3) movies based on average review socre.

For example, if the input movie id is “B0093ICOE0”, which is “Watchmen Collector’s Edition: Ultimate Cut + Graphic Novel [Blu-ray]”, then let’s see what we’ll get.

Finally, we have the results.

```
## [1] "Iron Man - Spanish Version"
## [2] "Eternal Sunshine Of The Spotless Mind (Eterno Resplandor De Una Mente Sin Rec
uerdos) [NTSC/REGION 1 & 4 DVD.Import-Latin America]"
## [3] "Collateral [Blu-ray]"
```

2.3 Word cloud of popular actors and directors

Wordcloud of Directors

Wordcloud of Actors

2.4 Recomendation Algorithm 2 & 3:

This recommendation algorithm based on biclustering. Intuitively, the results should share some similarities with the input (They are in the same cluster).

Advantage: 1. do simultaneous clustering of 2 dimensions 2. large datasets where clustering leads to diffuse results 3. only parts of the data influence each other

Disadvantage: 1. slow, but not the slowest one 2. not stable, need to tune parameters of the model

3 Chi-Squarred Method for Recommendation

3.1 χ^2 Applied to Common Reviewers

This recommendation method is based on the χ^2 statistic for independence of rows and columns in a contingency table. The main idea is to generate a contingency table based on common reviewers of two particular movies. This is constructed based on a binary variable **liked**, which can take the value L (liked movie) and N (not liked movie). An L is assigned for review scores equal to 5, and N is assigned otherwise. The contingency table is created by crossing this variable for movie i and movie j.

```
contingency.table <- table(liked.j,liked.i)
contingency.table
```

```
##        liked.i
## liked.j  L  N
##       L 13  5
##       N  4 11
```

```
chisq.test(contingency.table)$statistic
```

```
## X-squared
##  5.096599
```

The purpose of the χ^2 is to test the null hypothesis that rows are independent from columns. The statistic is computed by summing the squared differences between the observed and the expected count assuming independence. Therefore, under the assumption of independence the observed should be close to the expected and a small χ^2 should be observed. The following is an example of this:

```
contingency.table <- table(liked.j,liked.i)
contingency.table
```

```
##        liked.i
## liked.j  L  N
##       L  3  7
##       N  4 10
```

```
chisq.test(contingency.table)$statistic
```

```
##    X-squared
## 1.471656e-31
```

For cases with a strong row and column dependence, a high value of the χ^2 is expected. It is important to

note that strong dependence could have two meanings. The first one, that users who reviewed both movies agree most of the time, i.e., the majority either liked both or disliked both. The first example that was shown is a case of this. There is a second case of strong dependence where the users most of the time disagree, i.e., they liked one of the movies but disliked the other. The following is an example of this case:

```
contingency.table <- table(liked.j,liked.i)
contingency.table
```

```
##         liked.i
## liked.j  L  N
##       L  4  9
##       N 12  2
```

```
chisq.test(contingency.table)$statistic
```

```
## X-squared
##  6.306842
```

Note that both examples show a large χ^2 , however they have a different interpretation. For the purpose of this recommendation tool, only case one was considered. Even though case two is also meaningful, it was not considered in the sake of simplicity. The χ^2 was computed only for cases where the following is true:

$$LL + NN > NL + LN$$

3.2 Recommendation Tool

The recommendation tool is currently implemented as an R function, however it could easily be implemented in a shiny app. χ^2 values were computed as described before for each pair of movies in the training sample and saved in a data frame called PairChi. The user should give as a first input the movie id of one of their favorite movies. The second input is n, the number of recommnedations that want to be retreived. The function will extract from PairChi the χ^2 values corresponding to the input movie and the rest of the movies, they will be ranked and then a function will eliminate repetions (for example if there's a Titanic VHS and Titanic DVD only the first one will be considered). The first n movies will be recommended to the user. The following is an example of this:

```
return.name("B00004TX12")
chi.sqr.recommendation("B00004TX12",5)
```

```
## [1] "Watchmen Collector's Edition: Ultimate Cut + Graphic Novel [Blu-ray]"
```

```
##                                                 name2      chi
## 88                          Superman Returns [Blu-ray] 3.932664
## 107                            V for Vendetta [HD DVD] 2.387153
## 73  Sin City - Unrated (Two-Disc Collector's Edition) 2.343750
```

4 Model Evaluation and Recommendation Example

4.1 Train and Test Samples

A training sample of 40,589 observations, containing 403 movies and 2,481 users, was used to train all of the recommendation tools. A testing sample of 40,353 observations, containing 403 movies and 2,481 users, was used to evaluate our recommendation tools.

4.2 Model Evaluation Framework

The evaluation framework is based on a testing sample of 40,353 observations, containing 403 movies and 2,481 users . The following table is an example of the evaluation framework:

Recommendation Method 1

Input Movies	Recommended Movie	Average score (Testing sample)
Titanic	The Notebook	4.5 (computed based on people who liked Titanic and reviewed The Notebook)
Watchman	Superman	4.8
*	*	
*	*	
*	*	
Harry Potter	Lord of the Rings	
Average		**4.75**

This table is created for each method. The first column is all of the possible inputs from which the user can choose their favorite movie. The second column is the movie recommended by Method 1. To explain the third column let's focus on the first example. If the user chooses Titanic, the recommended movie would be The Notebook. From the test sample we take all the users who liked Titanic and also reviewed The Notebook and compute the average score they gave to The Notebook. Then the average of all of the rows is computed to get an overall score of the recommendation method. We do this for our three recommendation tools and compare.

4.3 Evaluaton Results

Recommendation Method	Average Score
Network	4.31
Biclustering	4.51
Chi-Square	4.46

The Biclustering and Chi-square methods are very close, however the biclustering is a little bit better.

4.4 Recommendation Example

This is an example of what the three methods would recommend if the input movie is “Watchmen movie” which is a movie based on a DC comic.

	recommendation_name	director	actor1	actor2	actor3	review_score	review_num
Network Analysis	Ratatouille	Brad Bird	Brad Garrett	Lou Romano	Patton Oswalt	4.48	162
	Iron Man - Spanish Version	Jon Favreau	Jeff Bridges	Jr. Robert Downey	Clark Gregg	4.47	267
	The Departed (Blu-ray/DVD Bundle)	NA	NA	NA	NA	4.21	289
Biclustering	Sweeney Todd - The Demon Barber of Fleet Street	Tim Burton	Helena Bonham Carter	Johnny Depp	Alan Rickman	4.20	155
	Live Free or Die Hard - Unrated (Two-Disc Special Edition)	Len Wiseman	Bruce Willis	Justin Long	Timothy Olyphant	4.09	170
	Watchmen	Zack Snyder	Malin Akerman	Billy Crudup	Matthew Goode	3.91	178
Chi^2 Analysis	Superman Returns [Blu-ray]	Bryan Singer	Brandon Routh	Kate Bosworth	Kevin Spacey	3.52	261
	V for Vendetta [HD DVD]	James McTeigue	Natalie Portman	Hugo Weaving	Stephen Rea	4.08	218
	Sin City - Unrated (Two-Disc Collector's Edition)	Frank Miller (II)	Jessica Alba	Devon Aoki	Alexis Bledel	4.18	298

There are some similarities between these movies. There are some “superheroes movies” recommended such as “Ironman” and “Superman”. We can also see that some are are action movies like “V for Vendetta”. Many of this movies can also be categorized as “Good vs Bad Guys” movies.